MASTER EN SCIENCES INFORMATIQUES

Optimisation d'un Processus de Production de Poutrelles par la Méthode du Tabou

Davin, Stéphane

Award date:
1994

Awarding institution:
Universite de Namur

Faculté Universitaire Notre-Dame de la Paix

Institut d'Informatique

Rue Grandgagnage, 21

B-5000 Namur

Optimisation d'un Processus de Production de Poutrelles par la Méthode du Tabou

Mémoire réalisé pour l'obtention du grade de

Licencié & Maître en Informatique

par

Davin Stéphane

Promoteur : Leclercq Jean-Paul, F.U.N.D.P.
Co-promoteur : Lescrenier Marc, ARBED Esch-sur-Alzette

Année académique 1993-1994

OBJET :

Ce document traite de l'étude de la méthode du Tabou lors de son application à un problème d'optimisation d'un processus de production de poutrelles.

Le problème sera abordé en trois étapes qui définiront chacune un modèle de difficulté croissante. Elles permettront de distinguer l'influence des élements de la méthode.

Le document présente tout d'abord les différentes filières de production du groupe Arbed et notamment de manière détaillée la filière électrique avec une coulée continue. La méthode du Tabou est ensuite décrite. Après, nous procéderons au développement complet des deux premiers modèles et à l'analyse détaillée des résultats du développement de ceux-ci. Enfin, une conclusion sur la méthode du Tabou sera faite.

Le document propose également une préétude du troisième modèle.

ABSTRACT :

This paper deals with the study of the Tabou method in its application to an optimisation problem of steel manufacturing.

The problem will deal within three steps which all define a model of croissant difficulty. They will allow to distinguish the influence of the method's elements.

This paper introduces first the different chanels of the Arbed group and in particular in details the electric chanel with a continuous casting. The Tabou's method is then described. Next, we will procede to the complete development of the two first models and to analysis of their development results in details. Finaly, a conclusion of the Tabou's method will be made.

An approach of the third model is also dealt with.

Remerciements

Je tiens à remercier toute l'équipe de l'Unité Modélisations & Optimisations pour l'accueil chaleureux, la disponibilité et la sympathie à mon égard lors de mes visites; Monsieur Raymond Bausch, responsable de l'Unité, à qui il a été agréable de présenter l'état d'avancement du projet; Marc Lescrenier, responsable du projet, pour la collaboration étroite au cours de laquelle un climat de confiance n'a cessé d'exister, ainsi que pour tous les conseils toujours judicieux.

Je remercie également Monsieur Jean-Paul Leclercq, promoteur de ce mémoire, pour me l'avoir proposé, pour le temps qu'il a pu me consacrer à la lecture de ce mémoire, pour sa critique judicieuse et ses conseils dans les différentes étapes de la rédaction ainsi que dans l'analyse du projet.

Enfin, je remercie toutes les personnes qui, de près comme de loin, m'ont soutenu lors de la réalisation de ce mémoire, et en particulier Angélique.

CHAPITRE I :INTRODUCTION

A. Composition du document.

Une introduction générale à tous les élements de ce mémoire constitue le premier chapitre. Au sein de ce chapitre seront abordés respectivement les points suivants : une présentation générale des filières de production de poutrelles du groupe Arbed et notamment la filière électrique avec une coulée continue, le problème et les différentes étapes de résolution de celui-ci, caractérisées chacune par un modèle, une présentation détaillée de la méthode du Tabou.

La réalisation du modèle 1 compose le second chapitre qui commencera par une description sous forme des inputs et des outputs du modèle. Ensuite, viendra le développement du modèle et l'analyse de la méthode du Tabou à partir des résultats du développement. Quelques conclusions sur la méthode du Tabou termineront ce chapitre.

Le troisième chapitre concerne le modèle 2. Il sera structuré de la même manière que le deuxième chapitre : description sous forme des inputs et des outputs, développement, analyse de la méthode à partir des résultats du développement, et, quelques conclusions sur la méthode du Tabou.

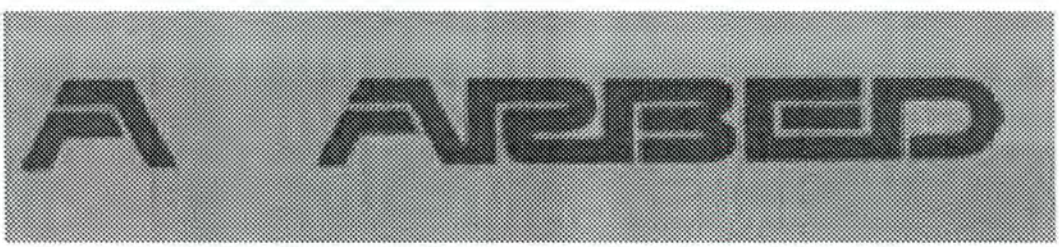
Vient ensuite le quatrième chapitre où nous tirerons les conclusions générales sur la méthode du Tabou.

Le cinquième et dernier chapitre introduit au troisième modèle.

B. Présentation sommaire de ProfilArbed.

a. Le groupe Arbed.

Constituée en 1882, ARBED S.A. Luxembourg est la société mère d'un groupe international) le groupe ARBED - composé de 10 secteurs d'activités et occupant 49.000 personnes dans le monde entier. Le chiffre d'affaires du groupe atteint LUF 190 milliards (ECU 4,7 milliards) en moyenne annuelle. Avec une capacité de production de 8 millions de tonnes d'acier par an, le groupe ARBED est le 6ème producteur sidérurgique en Europe et figure à la 15ème place sur la liste mondiale de l'Institut International du Fer et de l'Acier. Il occupe une position de leader sur le marché des poutrelles d'acier et des palplanches. Par sa filiale SIDMAR, en Belgique, il compte parmi les producteurs de tôles les plus performants d'Europe.



ARBED est le troisième producteur mondial de steelcord, et , par l'intermédiaire du groupe affilié Belgo-Mineira, au Brésil, le premier producteur de tréfilés du continent américain. Il est le seul producteur de tôles fines en acier inoxydable du Bénélux. Avec son partenaire japonais, Furukawa Electric, il figure à la troisième place des producteurs mondiaux de feuilles de cuivre extra-minces.

Le réseau de vente, de négoce et de trading compte une cinquantaine de points d'appui dans le monde entier.

Depuis sa création, Arbed est le plus important groupe industriel au Grand-Duché de Luxembourg, où il emploie plus de 12.500 personnes et contribue pour 10% à la formation du produit intérieur brut.

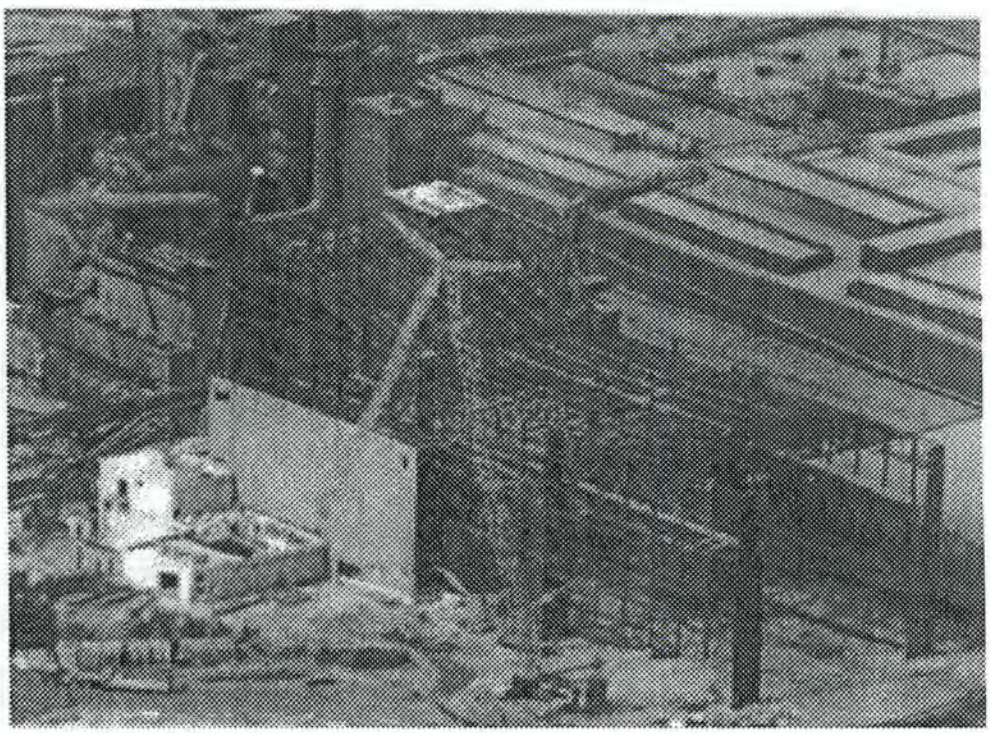
b. Les réalisations stratégiques et management de crise.

La crise actuelle n'épargne pas le monde de la sidérurgie où la diminution de la production mondiale d'acier de quelques 2,1 % (avec 3,7 en CEE) mais surtout la chute des prix de vente de 30 à 40 % (de 1990 à 1992) a constitué le problème fondamental de la sidérurgie en 1992.

Des adaptations indispensables doivent être réalisées à un moment où la sidérurgie industrielle doit faire face à une conjoncture affaiblie, marquée par la récession. Ainsi, par exemple, dans le secteur des produits longs qui contribue à presque 24% du chiffre d'affaire consolidé, la filiale ProfilARBED investit actuellement dans la construction d'aciéries électriques à Esch-Schifflange et Differdange, et, d'une coulée continue en amont du train Grey de Differdange.

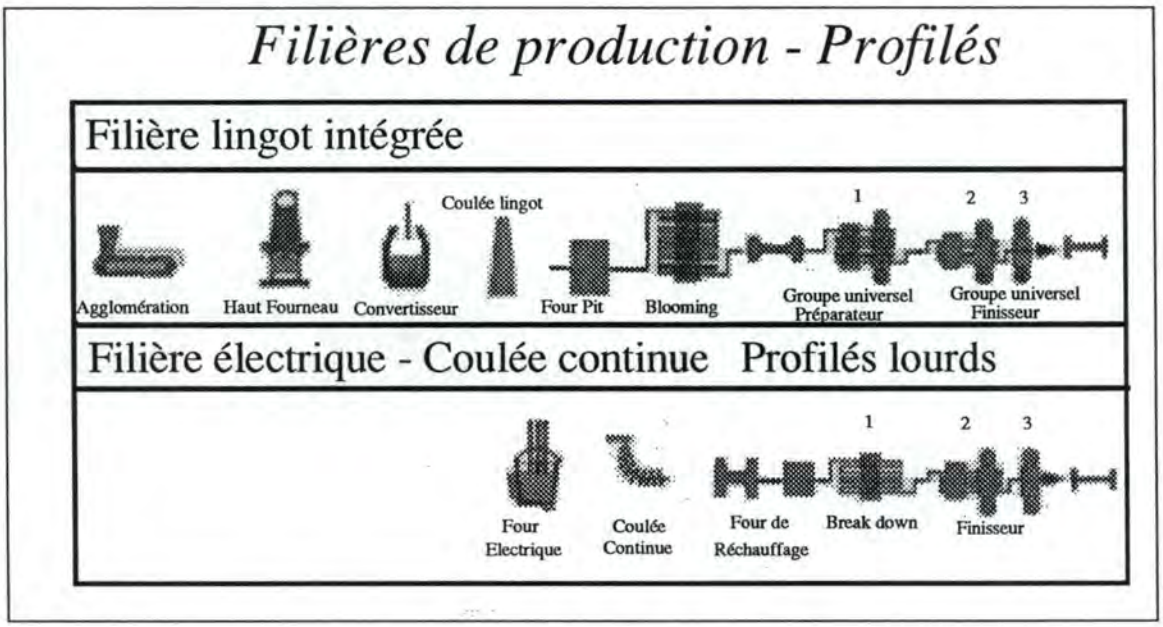
CHANTIER DE L'ACIERIE ELECTRIQUE D'ESCH-SCHIFFLANGE

(juillet 1993)



C. Présentation des processus de production de poutrelles - Les filières de production.

On peut distinguer deux types de filières : la filière fonte et la filière électrique. Chaque type de filière est lui-même divisé en deux sous-filières : la filière lingot et la filière coulée continue.



a. La filière fonte.

Dans la filière lingots, comme son nom l'indique, ce sont des lingots qui vont être mis au four avant d'être laminés; dans la filière coulée continue, ce traitement sera appliqué à des beams blanks qui sont en fait des ébauches de barres-mères.

FILIERE FONTE

Filière lingots	*Filière coulée continue*
Haut fourneau	Haut fourneau
Aciérie	Aciérie
Lingots	Coulée continue
Four	Four
Laminage	Laminage
Scies	Scies

Les deux paragraphes suivants vous présentent en détails ces deux filières.

a.1. La filière fonte lingots

La toute première opération de cette filière est l'extraction du minerai de fer. On s'accorde à dire que la matière extraite est du minerai de fer quand celui-ci possède une teneur en fer d'au moins 30 %.

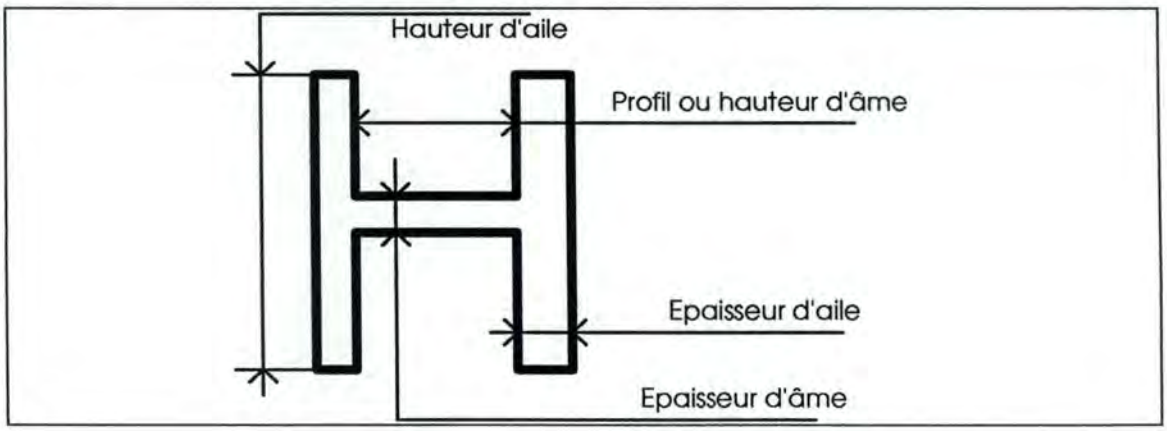
Une préparation de la charge s'impose pour que le haut fourneau remplisse sa tâche; celle-ci se compose d'une série de traitements tels que un concassage, un criblage, une agglomération et une homogénéisation.

La charge est ensuite conduite vers le haut-fourneau où elle est mélangée avec de la chaux, du coke, de l'air et un peu de mitraille (pour obtenir l'acier désiré). Le résultat est une fonte cassante que l'aciérie transforme en acier résistant.

Le travail de l'aciérie se fait en deux étapes : une première qui consiste à affiner la fonte c'est-à-dire à lui faire subir une oxydation de ses éléments (pour réduire la teneur en carbone par exemple) et une deuxième, que l'on nomme la métallurgie secondaire, où les traitements permettent d'arriver à la qualité d'acier désirée. La qualité de l'acier est aussi appelée ***nuance*** de l'acier.

L'acier est ensuite coulé dans des lingots qui ont une forme parallélépipédique dont la base est légèrement plus large que la partie supérieure. L'aciérie fournit l'acier en des séquences de coulées dont une seule permet le moule de plus ou moins 10 lingots (cela dépend du produit fini). Ensuite, une fois refroidis, les lingots sont démoulés puis introduits dans le four.

Une fois à bonne température, ils subissent une première opération de laminage qui se nomme blooming. Le résultat donne une ébauche de ***barre-mère.*** Celle-ci peut être laminée au travers de cages de laminage suivant le procédé Grey pour donner une barre-mère d'une centaine de mètres. Les cages de laminage ont donné à la barre-mère une forme bien précise. Celle-ci dépend entièrement des cylindres qui composent la cage de laminage (une forme de H ou de I par exemple pour des poutrelles).



La figure montre les différentes dimensions de la barre-mère hormis sa longueur de ± 100 mètres. La partie qui relie les deux "montants" du H s'appelle l'**âme** tandis que les deux "montants" s'appellent les **ailes.**

L'ensemble des trois mesures hauteur d'aile, épaisseur d'aile et épaisseur d'âme est appelé **dérivé.** Une large gamme de dérivés est possible pour chaque **profil** (hauteur d'âme).

Ensuite, ces barres-mère sont conduites aux scies où elles sont découpées en poutrelles de longueur désirée par les clients. La découpe en poutrelles s'accompagne presque toujours d'une chute qu'on voudrait assez petite. Ensuite ces poutrelles partent au refroidissoir puis à la dresseuse où elles subissent un léger redressement si nécessaire.

Chaque poutrelle est marquée pour le suivi des produits. Il existe aussi une série de finition tels que la perforation de trous pour des boulons et des rivets, la mise en peinture,... Toutes ces opérations se font sur stock. On peut enfin signaler des essais de contrôle tels que essai de traction , essai de résilience, essai de pliage, essai de dureté, essai d'emboutissage et un examen aux ultrasons.

Si les poutrelles sont expédiées directement après le laminage c'est-à-dire qu'elles sont mises sur wagon au sortir du laminage, on parle d'expédition directe, sinon elles sont mises en stock pour une expédition différée.

Remarque : Le Brevet QST

Il s'agit d'un traitement que les poutrelles subissent après le laminage dans le but d'améliorer les propriétés mécaniques de celles-ci (c'est-à-dire obtenir d'autres nuances plus résistantes). Celui-ci n'est pas chimique et porte le nom de quenching and self tempering (ou trempe et autorevenu). Le procédé consiste en une succession de refroidissements par eau et de réchauffements automatiques de l'intérieur vers l'extérieur. Ce procédé a permis de mettre au point une nouvelle génération de poutrelles (les poutrelles HISTAR) qui permettent des économies de poids substancielles.

a.2. Filière fonte coulée continue

Le déroulement est le même que la filière fonte lingots jusqu'à obtenir l'acier d'une certaine nuance . Ensuite, au lieu d'être moulé dans des lingots, l'acier est versé dans un répartiteur au fond duquel des trous permettent la répartition de l'acier sur les brins. A la sortie du brin, on découpe des beams-blanks en continu à l'aide de brûleurs.

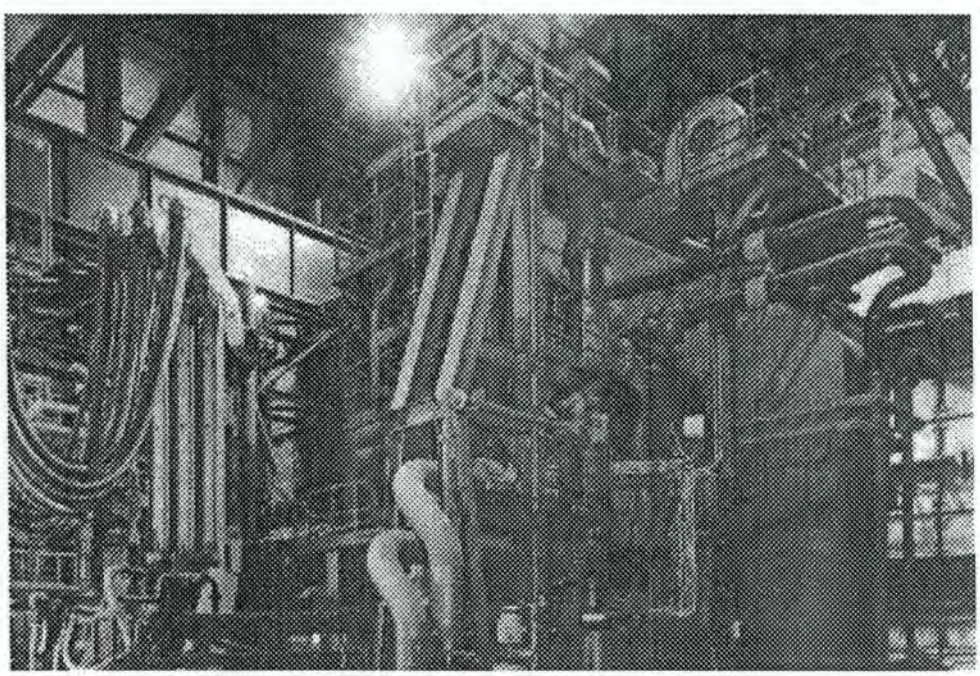
b. La filière électrique.

FILIERE ELECTRIQUE

Filière lingots	Filière coulée continue
Aciérie électrique	*Aciérie électrique*
Lingots	Coulée continue
Four	Four
Laminage	Laminage
Scies	Scies

C'est la partie supérieure du processus de production qui est différente par rapport à la filière fonte. Le haut-fourneau et l'aciérie sont remplacés par une aciérie électrique. Un four électrique produit les nuances d'acier à partir de mitrailles de récupération. La mitraille est l'élément essentiel pour la fabrication de l'acier comparativement à la filière fonte où celle-ci constituait un élément secondaire pour l'affinage de l'acier. Cette formule de production a l'avantage de supprimer le haut fourneau et l'extraction de minerai mais par contre le stock de la mitraille demande un espace important.

ESCH-SCHIFFLANGE : LE NOUVEAU FOUR ELECTRIQUE



Les deux types de filières électriques (filière à lingots et filière à coulée continue) se différencient de la même manière que pour la filière fonte.

ProfilARBED investit actuellement de manière à passer progressivement à la filière électrique et coulée continue. C'est précisément cette filière qui nous a intéressé dans ce stage. Pour être plus précis, le travail concernera la partie comprise entre la coulée continue et le laminage (voir la partie soulignée dans le schéma ci-dessus).

D. Présentation détaillée de la filière électrique avec une coulée continue.

a. Présentation détaillée de la filière électrique avec une coulée continue.

Elle se compose principalement de 6 éléments : l'aciérie électrique, la coulée continue, le four, le train de laminage, les scies et le stock de beams blanks.

L'aciérie électrique produit de l'acier à partir de mitrailles de récupération. En jouant sur divers paramètres qu'on n'explicitera pas ici, l'aciérie produit plusieurs qualités d'acier. Cet acier d'une qualité donnée (nuance) est coulé dans un répartiteur qui est une sorte de poche. Une coulée d'acier a un poids en acier bien défini et toujours le même. Ce répartiteur est percé de plusieurs orifices qui ont une taille bien définie en fonction du type de Beams Blanks que l'on veut produire. L'acier coule donc dans ces orifices et forme des brins. La coupe régulière à des longueurs précises (variant entre 8 et 12 mètres) de ces brins donnent des beams blanks.

Un répartiteur a une durée de vie limitée. Il ne peut servir qu'à un certain nombre de coulées maximum. A l'inverse, pour faciliter le travail de l'aciérie, le répartiteur doit servir à au moins un certain nombre de coulées d'une même nuance. Un répartiteur ne sert donc que pour un nombre de coulées compris entre ces deux bornes. Celles-ci dépendent entièrement de la qualité de l'acier ou nuance de l'acier.

L'ensemble composé du répartiteur et des moyens de découpe des brins compose la coulée continue. Celle-ci a une productivité constante qui ne dépend nullement de la nuance de l'acier coulé.

Les beams blanks produits par la coulée continue sont conduits soit vers le stock, soit vers le four. Le passage au four des beams blanks qui viennent soit de la coulée continue, soit du stock, dure entre 1 et 2 heures.

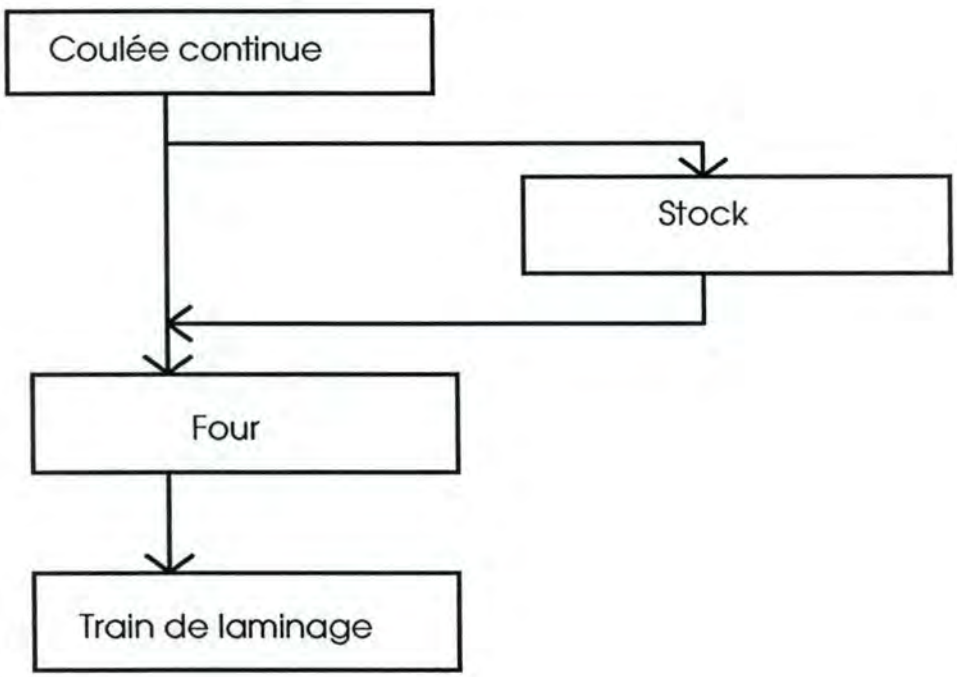
Ensuite, ils sont amenés vers le train de laminage. Ils sont alors transformés en barre-mère de $\pm$ 100 mètres de longueur. Enfin, ces dernières sont coupées aux longueurs désirées par les clients.

Le train lamine une large gamme de profils et dérivés. Il est bon de rappeler que le dérivé d'une barre-mère est l'ensemble des mesures suivantes : hauteur d'aile, épaisseur d'aile et épaisseur d'âme. Le profil représente la hauteur d'âme. Suivant que l'on lamine tel ou tel dérivé, le train fonctionne à des vitesses différentes.

b. Diagramme du processus de production.

La figure suivante nous montre le processus de production. Tous les éléments entrant dans ce processus ne s'y trouvent pas nécessairement. En effet, par exemple, le stock de poutrelles après le train de laminage ne s'y trouve pas. Il s'agit en fait des seuls éléments qui nous intéressent dans le cadre de notre problème.

DIAGRAMME DE PROCESSUS DE PRODUCTION



c. Organisation de la production.

ProfilARBED organise sa production de la manière suivante. Le point de départ est la commande du client. Celle-ci se compose en fait d'un ensemble de postes. Un poste correspond à x poutrelles de même nuance, même profil, même dérivé, même longueur.

Les commerciaux proposent une semaine de production au client pour chaque poste de commande. Ils le font en respectant les montages prévus (un montage =

1 profil laminé pendant 8 heures à telle date), et les disponibilités restantes (exprimées en tonnes) , ... et les délais souhaités par les clients.

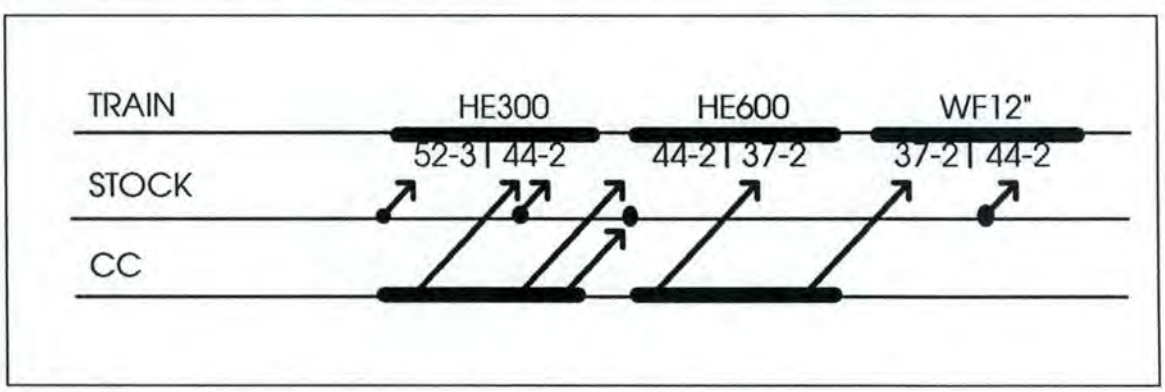
Pourquoi prévoir les montages des cylindres à l'avance et faire un même profil pendant 8 heures de suite ? Le temps de changement de cylindres pour des profils différents prend 2 heures. On ne peut donc pas s'arrêter toutes les heures pour changer de profil : le rendement serait très fortement réduit. On préférera alterner les dérivés avec même profil pendant 8 heures car le changement de dérivés se fait beaucoup plus rapidement. En résumé, on prévoit les montages pour chaque semaine et les postes de commandes sont affectés aux montages prévus par les commerciaux.

Le délai confirmé au client est donné par une semaine et non un jour précis de production.

d. Le problème.

D'un côté, le train lamine des barres-mère de même profil pendant 6 à 8 heures (un montage). Au sein de ce montage, il lamine des barres-mère de dérivés et de nuances variés. De l'autre, la coulée continue s'organise en séquences de coulées de même nuance.

Il nous faut donc, d'une part ordonnancer les séquences de coulées de la coulée continue et il nous faut aussi ordonnancer les montages et les dérivés au sein d'un même montage, dans le but de maximiser l'enfournement à chaud.



Ce problème va être résolu en 3 étapes. Au fur et à mesure de celles-ci, des éléments nouveaux vont être introduits pour arriver vers le problème complet mentionné ci-dessus. Chaque étape fera apparaître un modèle. Ceux-ci vont être spécifiés par la méthode des inputs - outputs dans la section suivante.

e. Les étapes de résolution.

Dans tous les modèles qui seront traités ici, l'objectif commun est de maximiser l'enfournement à chaud. Nous décrivons ici nos trois modèles sous forme de leurs inputs et outputs.

En utilisant le tonnage de beams blanks, nous introduisons une approximation par rapport au problème réel dans lequel c'est le nombre de beams blanks qui importe. Nous décrirons donc un stock ou un besoin de beams blanks par son tonnage nécessaire et non le nombre (et les dimensions) des beams blanks.

Modèle 1 :

Input :

- Une séquence (ordre) de montages pour une semaine avec heures de début de laminage, nuance, tonnage et productivité pour chaque dérivé.

- Un stock illimité de chacune des nuances.

Output :

- Les séquences de coulées à la coulée continue (nuance, nombre de coulées, heure de début)

Modèle 2 :

Input :

- Une séquence (ordre) de montages pour une semaine avec heures de début de laminage, nuance, tonnage et productivité pour chaque dérivé.

- Un état du stock limité pour certaines ou pour toutes les nuances.

Output :

- Les séquences de coulées à la coulée continue (nuance, nombre de coulées, heure de début)

- Un nouvel état du stock (limité).

Modèle 3 :

Input :

- Un ensemble de montages pour une semaine avec tonnage et productivité pour chaque dérivé.

- Un état du stock (limité)

Output :

- Les séquences de coulées à la coulée continue (nuance, nombre de coulées, heure de début)

- Les montages du train de laminage ordonnés (nuance, dérivé, tonnage, heure de début)

- Un nouvel état du stock (limité)

Remarque :

Nous venons de vous présenter les trois modèles pour résoudre le problème d'optimisation du processus de production de poutrelles. Seuls les deux premiers modèles font partie du cahier des charges. Afin de vous présenter le problème complet, nous vous avons aussi présenté le troisième modèle.

E. Introduction à la méthode du Tabou.

La méthode du Tabou est une méthode d'optimisation d'une fonction réelle f : $X \rightarrow \Re$. On cherche donc à minimiser cette fonction c'est-à-dire à trouver un $s^* \in X$ tel que $f(s^*)$ est minimum.

a. Une méthode descente.

La méthode du Tabou est basée sur la méthode de descente dont voici l'algorithme.

METHODE DE DESCENTE

initialisation — on choisit une solution initiale s appartenant à X

stop := faux

tant que — (stop=faux) faire

générer un ensemble V^* de solutions dans le voisinage de s

trouver la meilleure solution s' dans V^*

($f(s') = \min \{f(s^\circ) \mid s^\circ \in V^*\}$)

si ($f(s') \geq f(s)$) alors stop := vrai

sinon s := s'

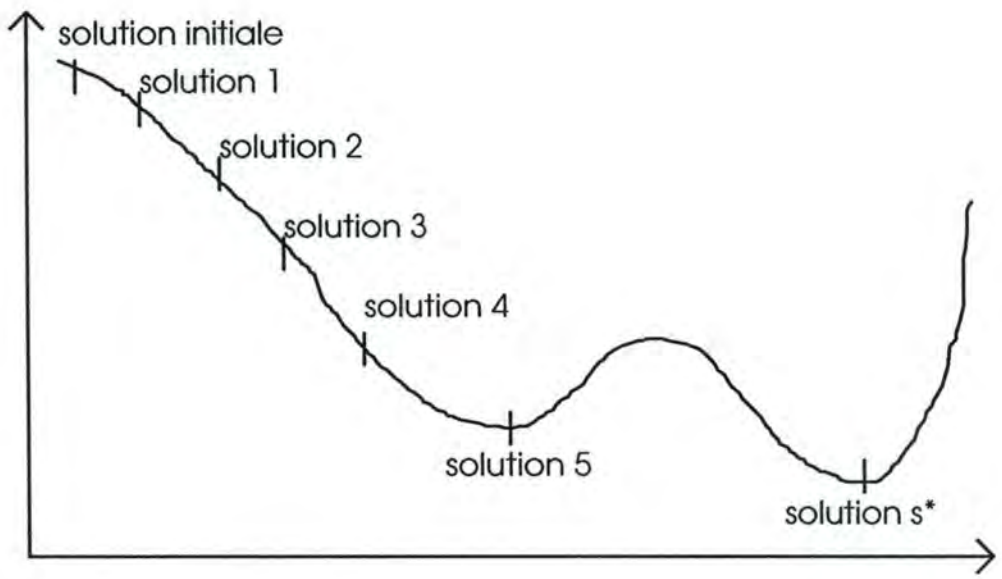
fin tant que

Dans cette méthode comme dans beaucoup de méthodes d'optimisation, la solution initiale ainsi que les solutions examinées au cours de la recherche appartiennent toutes à l'ensemble X. On génère donc un ensemble V* de solutions qui est inclu dans cet ensemble X. Une caractéristique fondamentale de

cette méthode est de reprendre la meilleure solution trouvée à l'itération précédente pour exécuter l'itération suivante.

Cette méthode court le risque grave de tomber dans un minimum local. Ainsi dans l'exemple suivant, on passe de la solution 1 à la solution 5 où l'algorithme s'arrête, malgré que le minimum se situe en s*.

EXEMPLE DE MINIMUM LOCAL



b. Principe général.

La méthode du Tabou essaye de pallier à ce problème de minimum local. Pour ce faire, on construit un point initial s_0 et on génère une suite s_0,s_1,s_2,s_3 $...,s_i,...,s_{n-1},s_n$ de points. Parmi tous ces points, un se distingue des autres par le fait qu'il minimise f c'est-à-dire $\exists!\ s^* : s^* \in (s_i)_{1 \le i \le n}$ et $\forall s_i : 1 \le i \le n : f(s_i) \ge f(s^*)$. Un critère d'arrêt permet de définir totalement cette suite. On espère que ce point-solution s* est une solution au problème de minimisation de f sur X c'est-à-dire que cette solution s* est telle que $\forall\ s \in X$, $f(s) \ge f(s^*)$, et, $s^* \in X$.

Le principe de la méthode tabou est le suivant. Une solution s peut subir une modification m pour donner une autre solution. De façon à ce que l'algorithme ait un degré raisonnable de simplicité, cette modification devra être restreinte à une classe de modification simple. Ainsi, nous définirons :

M ensemble des modifications acceptables

M_S ensemble des modifications acceptables à partir de s

La nouvelle solution est définie par : $s' = s \oplus m$, $m \in M$.

Le voisinage de s est l'ensemble des solutions accessibles par une modification à partir de s c'est-à-dire :

$$\mathbf{V(s) = \{s' \mid s' = s \oplus m \text{ et } m \in M_S \}}$$

Partant d'une solution s, une itération de la méthode du Tabou consiste à choisir un sous-ensemble V* de V(s). On détermine ensuite une solution s' minimale dans V* c'est-à-dire une solution s' telle que $f(s') = \min\{f(s) \mid s \in V^*\}$.

Remarque :

Le fait de déterminer le minimum dans le voisinage n'est pas toujours chose aisée. En général, on choisit soit de faire une optimisation locale, soit de restreindre l'ensemble des solutions du voisinage à un ensemble fini que l'on peut énumérer. En général, on choisira cette deuxième solution.

Si cette solution s' est meilleure que la solution optimale actuelle s*, cette dernière est remplacée par s', sinon cette solution s* ne change pas.

De toute façon, l'itération suivante (s'il y en a une) utilise s' comme solution courante.

c. La liste Tabou.

Comme nous l'avons vu, le piège des méthodes d'optimisation est de tomber dans un minimum local. Pour éviter cela, on doit accepter de "remonter" quelques fois, donc accepter des mauvaises solutions telles que : $s' \in V^* \subseteq V(s)$ et $f(s') > f(s)$. Mais cela peut impliquer un risque de bouclage. En effet, on passe d'une solution s à une solution s' qui appartient au voisinage de s. Mais alors, le voisinage de la solution s' comprend la solution s. Si cette solution est choisie, on bouclera.

Exemple :

Considérons le problème de coloration d'un graphe. Soit s une solution de coloration du graphe. Si l'on modifie la couleur d'un des sommets (soit a ce sommet) du graphe du rouge vers le bleu (cette nouvelle solution est la même que celle de départ excepté en son sommet a) et si l'on suppose que l'on a dégradé la fonction d'évaluation, on peut repasser vers la solution s qui appartient au voisinage de cette solution et ainsi boucler.

On introduit alors une liste T de solutions interdites ou taboues qui interdisent de revenir à une solution déjà visitée au moins pour $|T|$ itérations. Cette liste est circulaire c'est-à-dire que la solution taboue ou interdite la plus vieille sort de la liste au profit d'une nouvelle solution taboue. Le voisinage est alors l'ensemble des solutions que l'on peut atteindre par une modification moins les solutions contenues dans la liste taboue.

La choix de la taille de la liste est du ressort du modélisateur. Dans son problème de recherche de la plus grande clique dans un graphe (Référence [6]), Friden fixe la taille de sa liste Tabou à 2 sans aucune explication. Dans un autre article (Référence [7]), le même auteur fait varier la taille de la liste de 1 à 2 lorsque le problème a tendance à boucler. Pour le problème de coloration de graphe (référence [5]), Hertz fixe la taille de sa liste Tabou à la moitié du nombre de sommets de son graphe. Dans ces articles, il apparaît clairement qu'il n'y a pas de règle pour fixer la taille de cette liste : l'expérience et une bonne connaissance du problème sont probablement les meilleurs guides.

On est tenté de se munir d'une liste de grande taille pour être sûr d'éviter un bouclage. Mais, vu la taille en mémoire, parfois importante, d'une solution, cette liste peut occupé un volume mémoire inacceptable. Dans le problème de Hertz cité ci-dessus, une solution est un graphe qui peut prendre énormément de place lorsque le nombre de sommets est grand.

De plus, le test de présence d'une solution dans la liste peut prendre énormément de temps pour des solutions de taille importante et pour des listes taboues de grandes tailles. Toujours dans le problème de coloration de graphe de Hertz, le temps de comparaison de deux graphes peut prendre beaucoup de temps pour des graphes comportant parfois plusieurs milliers de sommets. Si en plus, la taille de la liste taboue est grande, il faudra multiplier ce temps par la longueur de la liste.

Afin d'éviter ces problèmes, on préférera définir un ensemble de conditions taboues et on attribuera un statut Tabou à une modification ou à une solution si elle satisfait à toutes les conditions taboues. Pratiquement, cela consiste à placer dans la liste taboue la modification qui vise à passer d'une solution à une autre, au lieu de la solution elle-même.

Cette approche ne sera intéressante que si la taille en mémoire de la liste des conditions taboues est inférieure à la liste des solutions. On espère que le temps de recherche sera lui aussi diminué.

Enfin, nous pouvons voir que cette approche donne un autre avantage : elle tente d'éviter de considérer des solutions qui sont différentes mais assez semblables à celles déjà visitées.

EXEMPLE DE LISTE TABOUE

s solution courante	m modification	T liste Tabou
ab	a → c	a → c
cb	b → a	a → c ; b → a
ca	c → b	b → a ; c → b

La présence de a → c dans la liste Tabou indique qu'il nous est interdit de faire la modification c → a.

La taille de la liste Tabou est de 2. C'est pourquoi la modification taboue a → c a "disparue" à la troisième itération, remplacée par la nouvelle modification c → b.

d. Condition d'aspiration ou fonction d'aspiration.

La méthode, telle que décrite jusque maintenant, n'est pas encore parfaite. En effet, il subsiste un risque de bouclage (par exemple, on peut revenir à une solution déjà visitée si la liste taboue est trop courte). Il peut exister des conditions taboues "illogiques" (on interdit d'envisager une solution qui serait intéressante de visiter). L'exemple suivant illustre parfaitement ces deux problèmes.

EXEMPLE DE RISQUE DE BOUCLAGE

s solution courante	m modification	T liste taboue
{a,b,c}	c → d	c → d
{a,b,d}	b → c	c → d ; b → c
{a,c,d}	d → b	c → d ; b → c ; d → b
{a,b,c}		

La modification d → b n'est pas interdite. Elle nous fait passer de la solution {a,c,d} à la solution {a,b,c}. Or, cette solution a déjà été visitée. Nous sommes bien en présence d'un bouclage.

EXEMPLE DE CONDITION ILLOGIQUE

s solution courante	m modification	T liste taboue
ab	b → c	b → c
ac	a → d	b → c ; a → d
dc	c → e	b → c ; a → d ; c → e
de		

Lorsque nous avons la solution s = {d,e}, nous observons que la liste T interdit la modification d → a de telle façon qu'il nous est impossible d'atteindre la solution s' = ae. Or, cette solution n' a pas encore été visitée. Nous sommes bien en présence d'une condition taboue "illogique".

On introduit alors un nouvel élément qui est un ensemble de conditions d'aspiration. Le principe est simple. Si le mouvement est autorisé - c'est-à-dire non tabou - , on peut passer à la solution via le mouvement. Si le mouvement est tabou et que la ou les condition(s) d'aspiration sont satisfaite(s), le passage à cette solution via le mouvement tabou est autorisé. En fait, ces conditions d'aspiration veulent nous donner une appréciation de la nouvelle solution obtenue par modification, malgré que cette dernière soit taboue.

En général, on compare la fonction d'évaluation prise en la nouvelle solution avec une valeur donnée par une fonction (d'aspiration). Il existe de nombreux exemples de celle-ci mais la plus courante est la fonction d'aspiration qui donne la valeur minimum atteinte par une solution que l'on a visitée. Ainsi, on accepte un mouvement tabou s'il permet d'améliorer la meilleur solution trouvée.

Dans Référence [3] , de Werra nous donne plusieurs exemples dont notamment celui explicité ci-dessus. Parmi ceux-ci, une condition d'aspiration est : "il faut gagner plus que ce qu'on a jamais gagné lors d'une modification". Il s'agit de mesurer l'importance de la modification. Dans Référence [6] , il s'agit de trouver la plus grande clique d'un graphe. La condition d'aspiration veut que cette nouvelle solution fasse grandir la taille de la clique d'au moins une unité par rapport à la plus grande clique trouvée. C'est en fait la même condition que la fonction explicitée dans le paragraphe ci-avant. de Muynck utilise aussi la comparaison de la meilleure solution trouvée avec la nouvelle solution. Il ressort de ce petit tour bibliographique que la condition d'aspiration la plus usitée est celle qui lève le statut tabou à une solution lorsqu'elle donne une meilleur valeur que celle qu'on avait trouvée jusqu'alors.

e. Solution réalisable et pénalité.

Dans bien des méthodes d'optimisation, toutes les solutions générées au cours de la méthode appartiennent à l'ensemble X des solutions. La méthode du Tabou se distingue de celles-ci par le fait qu'aucune condition sur l'appartenance ou non des solutions n'est formulée, exception faite de la solution optimale s* fournie à la fin de la méthode. Il est bien dit "à la fin" car cette solution optimale s* peut ne pas appartenir à X dans la plus grande partie de la méthode.

Cette caractéristique peut amener dans certains cas à introduire un système de pénalités. Celui-ci vise essentiellement à faire revenir les solutions générées dans l'ensemble X des solutions acceptables afin de pouvoir fournir une solution appartenant à X à la fin de la méthode. Mais ce système autorise de visiter des solutions non réalisables qui pourraient être des intermédiaires obligés entre deux solutions réalisables lorsque, par exemple, le domaine de solutions est non connexe. Il reste le choix de la valeur de la pénalité qui peut être soit fixe, soit

variable d'une itération à une autre. Celui-ci est encore du ressort du modélisateur qui doit avoir une bonne connaissance du problème.

Exemple :

Le problème de la coloration de graphe nous donne un domaine de solutions qui est non connexe. Celui-ci peut nous amener à visiter uniquement des solutions non acceptables; seule la solution finale serait acceptable.

A ce stade, nous avons décrit les caractéristiques essentielles de la méthode de recherche taboue. Vous trouverez ci-dessous une description de l'algorithme *général*. Toutes variantes ou restrictions décrites ci-avant peuvent y être introduites.

ALGORITHME GENERAL DE LA METHODE TABOUE

initialisation

construire une solution initiale s

s* := s

nbiter := 0

bestiter := 0

initialiser la liste taboue T

initialiser les fonctions d'aspiration A à la valeur $+\infty$

kmax = nombre maximum d'itérations entre 2 améliorations de s*

tant que ((nbiter-bestiter) < kmax) faire

nbiter := nbiter + 1

générer un ensemble $V^* \subseteq N(s)$ de solutions š=s⊕m telles que

a) š ne viole pas les conditions taboues

b) š viole une condition mais les conditions d'aspiration sont remplies

choisir la meilleure solution s' dans V*

si f(s') < f(s*) alors s* := s' ; bestiter := nbiter

mettre à jour les listes taboues

mettre à jour les valeurs des fonctions d'aspiration

s := s'

fin tant que

F. Ingrédients de la méthode du Tabou.

Borne inférieure. [Référence n° 3,5]

Dans notre algoritme standard, si on n'améliore pas notre meilleure solution trouvée pendant Kmax itérations alors on s'arrête.

Dans certaines applications, nous connaissons une borne inférieure f* à la fonction f tel que $\forall$ s : f(s) $\geq$ f*. On peut arrêter d'itérer lorsque la valeur courante de la meilleure solution trouvée s* donne une valeur f(s*) qui est assez proche de f*.

Exemple :

Si nous essayons de construire un ensemble indépendant S de taille k dans un graphe, on définit normalement X comme une collection de tous les sous-ensembles S de sommets avec $|S| = k$. La fonction objective $f(s=S) = |E(S)|$ est simplement le nombre d'arètes du graphe qui ont leurs 2 sommets à la fois dans S. Alors, f*=0 est une borne inférieure pour f et S sera un ensemble indépendant si et seulement si f(s)=0=f*.

Système de poids. [Référence n° 3,5]

1) Supposons que nous devons trouver un ensemble indépendant de taille maximum dans un graphe. Nous allons essayer de prendre X comme la collection de tous les ensembles indépendants et définir $f(s=S) = |S|$. Une telle formulation n'est pas bonne. En effet, une des raisons est que les modifications m sont très difficiles à définir et elles peuvent prendre beaucoup de temps. Une façon plus habile est de considérer X comme l'ensemble de tous les sous-ensembles de sommets et d'introduire alors dans f un terme qui pénaliserait un ensemble S qui n'est pas indépendant. Par exemple, nous pouvons prendre $f(s=S) = -|S| + \alpha |E(S)|$ où α est un paramètre positif qui peut varier si nécessaire. Ainsi, la modification m peut alors être choisie de façon simple. Par exemple, nous pouvons définir une modification comme le mouvement d'un sommet de S vers G-S où G est l'ensemble des sommets du graphe.

2) Lorsqu'une région de X semble contenir de bonnes solutions, la recherche devrait être intensifiée dans cette région. Ceci peut être fait en augmentant artificiellement un peu f pour toutes les solutions qui ne sont pas dans la région paraissant intéressante.

Exemple :

Dans le problème du voyageur de commerce, supposons que nous ayons déjà construit beaucoup de solutions jusqu'à cette itération. Alors, nous pouvons remarquer que les bonnes solutions obtenues ont des propriétés communes. Par exemple, il y a un ensemble F d'arètes qui n'ont jamais été utilisées dans les solutions antérieures. Nous éliminerons alors ces arètes des candidats à l'inclusion ou à l'exclusion (= modification). Cela peut être fait en donnant un poids infini aux arètes de F. Durant quelques itérations, nous explorons le sous-ensemble correspondant des solutions avec le Tabou.

Exploration en profondeur de X. [Référence n° 3]

Une technique de recherche intelligente devrait non seulement exploré une région qui contient des bonnes solutions mais devrait aussi avoir une vue générale sur l'ensemble X et être sûre que des régions éloignées n'ont pas été négligées.

Une façon simple d'approcher ce but est de répéter la recherche toute entière avec une collection de solutions initiales générées aléatoirement. Nous pourrions alors tirer quelques conclusions statistiques intéressantes. Dans l'esprit de la recherche Tabou, il nous semble plus raisonnable de générer quelques solutions initiales qui ne sont pas aléatoires mais choisies précisément dans quelques régions de X qui n'ont pas encore été explorées plus tôt.

Exemple :

Toujours dans la recherche d'un ensemble indépendant de taille maximum dans un graphe, nous pouvons pénaliser les sommets qui ont été le plus souvent inclus dans les solutions visitées dans les itérations précédentes. Pour réaliser ceci, nous pouvons prendre : $f'(s=S) = |E(S)| + \alpha \Sigma_{i \in S} w_i$ où w_i est le nombre de solutions visitées plus tôt qui contiennent le sommet i.

Ingrédient pour le caractère tabou d'une solution. [Référence n° 7]

Il peut paraitre trop restrictif d'attribuer un statut tabou en se basant sur une liste. C'est pourquoi le caractère tabou pourrait dépendre aléatoirement d'une loi uniforme sur [0,1]. C'est-à-dire que l'on comparerait une probabilité p fixée (ou variable) avec un nombre choisi aléatoirement entre 0 et 1.

On peut adapter la probabilité et ainsi orienter le parcours du Tabou. Ce procédé nous semble quelque peu difficile à réaliser dans un problème sans une expérimentation qui peut être longue et fastidieuse.

Exemple :

Soit p une probabilité (un nombre entre 0 et 1) et r un nombre choisi aléatoirement entre 0 et 1. Une solution est taboue si elle appartient à la liste et si la condition $r \leq p$ est vérifiée.

Utilisation consécutive du Tabou. [Référence n° 5,6]

Nous avions déjà dit que le Tabou pouvait être utilisé plusieurs fois consécutivement en prenant des solutions initiales différentes. Mais il existe un autre type d'ingrédient qui procède de la même façon.

Pour des problèmes dont la taille de la solution est inconnue, nous choisissons une taille que nous pensons être la plus grande possible et nous faisons plusieurs optimisations par le Tabou sur des tailles décroissantes jusqu'à obtenir une solution dans une des optimisations.

Exemple :

Reprenons encore le problème de l'ensemble indépendant. Nous ne connaissons pas la taille maximum de celui-ci. Nous savons que le plus grand ensemble indépendant ne saurait être plus grand que la taille du graphe. Nous faisons une suite d'optimisations par le Tabou qui recherche un ensemble indépendant de taille décroissante d'une optimisation à une autre.

Des voisinages de plusieurs types. [Référence n° 5]

Dans certains problèmes, il peut être intéressant de développer des voisinages de plusieurs types. Le choix de celui-ci peut être fait de plusieurs manières : aléatoirement, alternativement,...

Exemple :

Le problème de la coloration de graphe consiste à attribuer une couleur à chaque sommet de telle façon que deux sommets reliés par une arète soient de couleurs différentes et que le nombre de couleurs utilisées soit le plus petit possible. Nous considérons que l'ensemble des sommets est partitionné en sous-ensemble S_i de même couleur et que l'ensemble des solutions X est l'ensemble des partitions de l'ensemble des sommets.

Nous pouvons considérer deux types de modifications : la première déplace un sommet d'un sous-ensemble S_i vers un sous-ensemble S_j, la deuxième déplace un sommet d'un sous-ensemble S_i vers S_j avec $i \neq j$. Ce deuxième de modification est intéressant si l'on veut fixer la cardinalité. On peut voir les premières itérations du Tabou faites avec le premier type de voisinage et les dernières avec le deuxième type de voisinage. On peut aussi les mélanger c'est-à-dire que d'une itération à une autre, on peut changer de type de voisinage.

CHAPITRE II : LE MODELE 1

A. Enoncé input - output du problème du modèle 1.

L'énoncé du problème du modèle 1 peut se résumer comme suit : étant donné la production de montages au train Grey définie par une séquence de ($nuance_i$,$tonnage_i$), trouver l'ordre et les séquences des coulées à prévoir afin de maximiser l'enfournement à chaud.

Ce problème peut paraître simpliste mais le but de ce modèle 1 est plus de mettre l'accent l'accent sur les qualités et les défauts de la méthode du Tabou que sur le problème en lui-même. On espère quand même qu'il servira de base pour l'élaboration du modèle 2. En fait, une méthode annexe (la programmation dynamique) nous permet de connaître la solution exacte (l'optimum) de ce problème. On pourra ainsi tester la validité de la méthode du Tabou sur un problème dont on connaît la solution.

On remarque immédiatement les simplifications de ce premier modèle. D'une part, le stock des nuances n'entre pas en jeu. En fait, on peut le considérer comme présent mais ayant une valeur infinie pour chaque nuance.

D'autre part, l'ordonnancement des montages (profils) et des nuances est donné. Dans la séquence des montages au train qui est une donnée, un montage apparaît comme l'intervalle entre deux pauses nécessaires au changement de cylindre. Les dérivés n'apparaissent plus explicitement dans les données. Ils apparaissent par la productivité associée à chaque nuance.

Input :

Le stock.

Une suite de nuances avec pour chacune d'elles, un stock illimité de cette nuance, une borne inférieure et une borne supérieure, ces deux bornes sont les limites d'utilisation du répartiteur pour cette nuance d'acier, c'est-à-dire :

$$(Nuance_i\ ,\ Stock\_nuance_i\ ,\ Borne\_inf_i\ ,\ Borne\_sup_i\ )_{1 \le i \le Nbr\_nuance}$$

où :

1) Nbr_nuance est le nombre de nuances que la coulée continue peut fournir.

2) $Stock\_nuance_i = +\infty\ ,\ \forall i : 1 \le i \le Nbre\_nuance$

La production au train.

Une séquence de montages pour une semaine avec heures de début de laminage, tonnage et productivité c'est-à-dire :

$$\text{Train} = \{(\text{Tonnage}_i , \text{Productivité}_i , \text{Nuance}_i , \text{Heure de début}_i)_{1\leq i\leq lg\_tr}\}$$

où lg_tr est le nombre total de tonnages à produire, chaque tonnage correspondant à une nuance et une productivité pour ce tonnage.

Output :

La coulée continue.

Les séquences de coulées à la coulée continue avec nuance, nombre de coulées et heure de début de coulée c'est-à-dire :

$$\text{C.C.} = \{(\text{Nbre de coulées}_k , \text{Nuance}_k , \text{Heure de début}_k)_{1\leq k\leq lg\_cc}\}$$

où : 1) lg_cc est la longueur de la liste des séquences de coulée.

2) $\text{Nbre de coulées}_k \in [\text{Borne\_inf}_{\text{Nuance}_k} , \text{Borne\_sup}_{\text{Nuance}_k}]$

Remarque :

1°) La période de prodution du train ("Temps_total_du_train") est, en général, dans la filière que nous optimisons, de 5 jours par semaine. Dans un esprit de généralisation du modèle, ce temps peut être supérieur ou inférieur à ces 5 jours. La coulée continue produit pendant 6 jours consécutifs. Ainsi, deux semaines de production font 10 jours de production pour le train et 12 jours de production pour la coulée continue.

$$\Sigma_{i\leq lg\_tr} (\text{tonnage}_i/\text{productivité}_i) = \text{Temps\_total\_du\_train}$$

2°) Dans l'introduction, il a été fait mention de la présence de pauses entre les montages au train. Nous expliquerons plus loin la modélisation de celles-ci dans notre problème.

B. Exemple numérique du modèle 1.

Nous allons vous présenter le problème du modèle 1 sous ces inputs et outputs représentés par des tableaux de valeurs. Ainsi, nous retrouvons le train décrit par le tableau "la production du train" et les nuances présentes auquelles sont associées les limites d'utilisation du répartiteur pour cette nuance. Ensuite, le résultat de notre algorithme est présenté : le tableau des séquences de coulées. Vous trouverez représentés les zones d'enfournement à chaud liées à la coulée continue et au train décrits dans les tableaux. Et enfin, une idée de la taille de l'ensemble des solutions nous permet de considérer la difficulté du problème.

Input :

La production du train :

Tonnage	Productivité	Nuance
2000	150	04
1500	200	09
1200	200	10
1000	200	02
800	150	09
3000	250	04
1500	150	02
450	100	10
80	40	01
400	100	09
150	150	04
600	150	02
450	100	10
4000	250	09
1200	120	04
350	175	01
90	90	02
2500	200	10
850	250	04

Le stock :

Nuance	Qté	B. I.	B. S.
1	9999	8	11
2	9999	8	11
3	9999	8	11
4	9999	8	11
5	9999	8	11
6	9999	8	11
7	9999	8	11
8	9999	8	11
9	9999	8	11
10	9999	8	11
11	9999	8	11
12	9999	8	11
13	9999	8	11
14	9999	8	11
15	9999	8	11

Période de production du train : 124 h
Qté : Quantité en stock
B.I. : Borne inférieure sur le nombre de coulées consécutives de la nuance pour un répartiteur
B.F. : Borne supérieure sur le nombre de coulées consécutives de la nuance pour un répartiteur

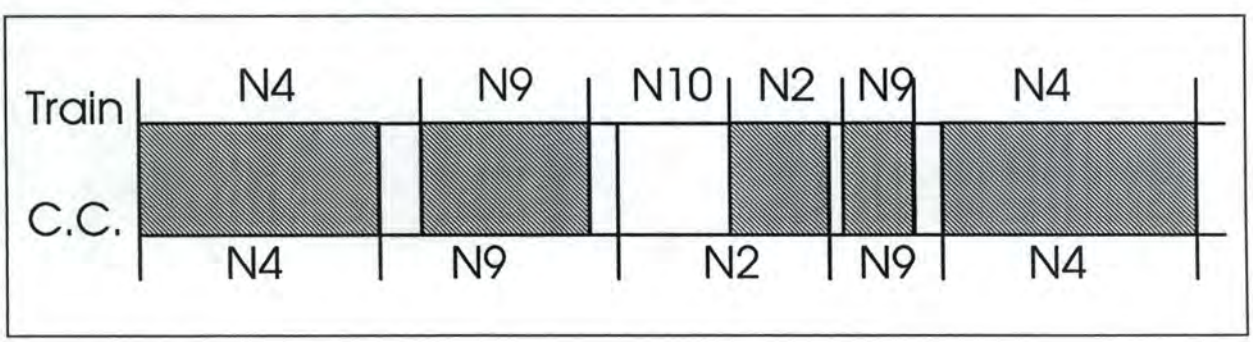
Output :

Tonnage à chaud : 12972 t

Séquence de coulées :

Nombre de coulée(s)	Nuance
10	4
11	9
9	2
8	9
11	4
11	2
10	9
9	2
11	9
9	9
11	4
11	10
9	4

Voici une représentation de la production du train de laminage et de la coulée continue (C.C.). Les parties hachurées correspondent à de l'enfournement à chaud. Il faut remarquer qu'en réalité, l'origine du temps pour le train et pour la coulée continue ne coïncident pas mais on les fait coïncider pour pouvoir voir directement les zones d'enfournement à chaud.



Chaque séquence de coulées est caractérisée par :

1) un nombre de coulées : 8 à 11 → 4 possibilités

2) un numéro de nuance : 1 à 15 → 15 possibilités

Il y a au minimum 15 séquences de coulées pour une semaine de 5 jours. En effet, 15 séquences de coulées avec 8 coulées par séquence font 120 heures c'est-à-dire 5 jours x 24 heures. Les 8 coulées pour une séquence sont le minimum que l'on puisse mettre comme nombre de coulées (voir les tableaux ci-avant).

Nombre de solutions possibles pour chaque séquence : 4 x 15 = 60 possibilités

Nombre total de solutions possibles : 60^{15} = 2 170 000 000 000 000 000 000 solutions possibles

Toutes ces solutions, même sur le plus grand des ordinateurs, ne pourraient être générées en vue d'être testées une à une. C'est pourquoi une méthode d'optimisation est nécessaire.

C. Calcul de l'enfournement à chaud.

L'objectif du problème est de maximiser l'enfournement à chaud. Il nous faut donc bien comprendre ce que cela représente. Quelques rappels, succincts mais essentiels, sur le processus de production des poutrelles nous éclairera sur ce problème. Ensuite, nous expliquerons la procédure de calcul de l'enfournement à chaud et sa formalisation mathématique.

Comme vu précédemment, l'aciérie électrique coule de l'acier dans un récipient appelé répartiteur. Dans celui-ci, trois orifices laissent passer l'acier qui coule en formant trois brins. Ces derniers sont coupés en des longueurs voulues. Le résultat en est des beams blanks. Ceux-ci ont alors deux destinations possibles : le four ou le stock. La décision de destination est prise en fonction des besoins au train de laminage.

Si le beams blank est conduit au stock, il y reste jusqu'à ce qu'on vienne le chercher pour le mettre dans le four. Cette opération est appelée enfournement à froid.

L'autre opération qui vise, au sortir de la coulée, à conduire directement le beams blank vers le four est appelée l'enfournement à chaud. C'est cela qu'il faut favoriser au maximum. Il faut que pour le plus grand nombre de beams blanks possibles, la direction choisie après la coulée continue, soit le four.

Les deux éléments nécessaires pour calculer l'enfournement à chaud, sont les besoins sur le train et l'acier produit à la coulée continue. Il nous faut trouver des zones où la coulée continue produit ce que le train "réclame". Une première condition pour avoir cela est que l'acier produit par la coulée continue soit de même nuance que celui dont le train a besoin. Cette condition est même nécessaire et suffisante pour avoir de l'enfournement à chaud.

Mais, on ne peut pas en déduire directement une quantité d'enfournement à chaud. En effet, les productivités du train et de la coulée continue ne sont pas nécessairement les mêmes. Si la coulée produit plus vite que le train ne lamine, une partie de l'acier ira vers le four mais l'autre ira vers le stock. Donc, toute la quantité produite ne constitue pas de l'enfournement à chaud. De même, si le train lamine plus vite que la coulée continue ne peut couler, une partie sera prise sur le stock et mise au four pour satisfaire les besoins en beams blanks du train de laminage. D'où pour une période donnée, l'enfournement à chaud est le minimum entre ces deux valeurs, à savoir, la quantité d'acier produite à la coulée continue et la quantité d'acier qui doit être laminée au train.

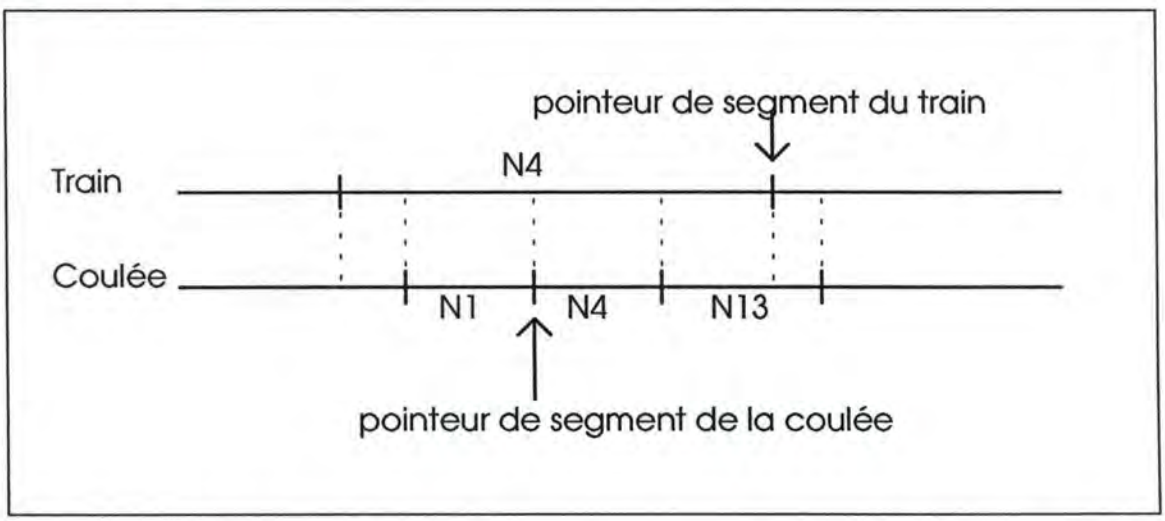
Le train ne peut laminer directement l'acier qui a été produit par la coulée continue. Il faut que celui-ci passe une à deux heures par le four. Pour pouvoir calculer l'enfournement à chaud, nous décalerons donc l'échelle du temps sur

laquelle est représenté le train, au niveau de l'échelle du temps où est représenté la coulée continue. L'avancement du train dans le temps sera égal à la durée de séjour des beams blanks dans le four. Donc, si nous faisons correspondre les origines du train et de la coulée continue, il nous reste à comparer les zones communes.

En conclusion, il nous faut donc observer des intervalles de temps pour lesquels les nuances de l'acier sont les mêmes au train et à la coulée continue. Ensuite, il nous faut prendre le minimum entre la quantité produite à l'un et la quantité laminée à l'autre.

Nous aurons donc deux droites segmentées dont les origines correspondent. Il nous faut comparer les segments de l'une avec les segments de l'autre. Un segment est caractérisé par une heure de début, une productivité, un certain tonnage et une nuance d'acier. L'heure de fin est évidemment donnée par le calcul suivant : pour le train, ajouter à l'heure de début le résultat de la division du tonnage par la productivité ou encore, pour la coulée continue, consulter l'heure de début de la prochaine coulée.

L'idée est d'avoir un pointeur de segment sur les deux droites et d'avancer progressivement dans le temps en comparant les segments. Nous comparerons le train et la coulée continue dans l'intervalle désignés par les deux pointeurs de segments. Il se peut qu'entre les temps désignés par ces deux pointeurs, il se trouve plusieurs segments. Dans le dessin suivant, on voit qu'il y a deux segments entiers de la coulée continue qui sont compris dans l'intervalle d'un segment du train.



Le segment de nuance N4 de la coulée continue englobe 3 segments du train respectivement de nuance N1, N4, N13. On comparera d'abord le segment de nuance N1 du train avec le segment N4 de la coulée continue. Ensuite, on avancera le pointeur de segment le moins avancé : celui du train. On comparera le segment de nuance N4 du train avec le même segment de la coulée continue. On

remarquera ensuite que le segment suivant de la coulée continue est à cheval sur celui du train. On comparera le bout de segment du train compris entre son début et la fin du segment de la coulée avec le bout de segment de la coulée compris entre les mêmes repères de temps.

On voit donc que le calcul de l'enfournement à chaud n'est pas aussi direct qu'on pourrait le croire surtout avec les deux problèmes cités juste ci dessus. Ces problèmes dépendent évidemment des données du train et de la coulée continue.

Le laminage au train dure 5 jours tandis que la coulée continue travaille 6 jours. On s'arrêtera de comparer le coulée continue et le train à la fin de ce dernier car plus aucune zone commune ne pourrait être trouvée dans ce 6ème jour où le train ne lamine pas.

Nous pouvons encore exprimer l'enfournement à chaud plus formellement de la façon suivante. A cette fin, nous allons introduire 5 définitions.

Définition 1 : [ZONE]

Une zone est un intervalle de temps (A,B) tel que :

1) Au temps A et au temps B, soit la coulée continue commence une séquence de coulées, soit le train commence un laminage.

2) entre A et B, aucun laminage d'une nouvelle nuance et aucune nouvelle séquence de coulées ne commence.

Outre les deux bornes A et B, cette zone est caractérisée aussi par la nuance Ncc et la productivité Pcc de la coulée continue, et, par la nuance Nt et la productivité Pt du train.

Exemple :

Dans le dessin précédent des deux droites segmentées, une zone est délimitée par les droites verticales en pointillées. On peut remarquer qu'il y a 4 zones pour un seul segment de la coulée continue. On doit donc bien distinguer ces deux éléments.

Définition 2 : [ZONE D'ENFOURNEMENT A CHAUD]

Une zone d'enfournement à chaud est une zone où la nuance Ncc de la coulée continue et la nuance Nt du train sont les mêmes.

Autrement dit, c'est une zone où la nuance réclamée par le train est celle produite par la coulée continue.

Définition 3 : [ZONE D'ENFOURNEMENT A FROID]

Une zone d'enfournement à froid est une zone où la nuance Ncc de la coulée continue et la nuance Nt du train sont différentes.

Donc, une zone est soit une zone d'enfournement à chaud, soit une zone d'enfournement à froid.

Définition 4 : [VALEUR DE L'ENFOURNEMENT A CHAUD]

Soit Z une zone d'enfournement à chaud.

Alors l'enfournement à chaud sur cette zone est :

$$\mathbf{EAC\ (Z) = \min\ (Pcc,Pt)\ .\ (B\text{-}A)}$$

Nous avons parlé plus haut des droites segmentées représentant la coulée continue et le train. Elles définissent une séquence finie de zones $(Z_i)_{1 \leq i \leq \#zones}$.

Définition 5 : [ENSEMBLE DES ZONES D'ENFOURNEMENT A CHAUD]

$ZAC = \{ Z_i \mid Z_i$ est une zone d'enfournement à chaud $\}$

L'enfournement à chaud pour une coulée continue et un train qui définissent une séquence de zones (Z_i) est :

$$\mathbf{EAC = \Sigma_{Z_i \in ZAC}\ EAC\ (Zi)}$$

D. Présentation avantage(s) - inconvénient(s) des structures de données possibles.

Nous allons passer en revue les éléments essentiels au problème et nous allons leur donner une structure de représentation possible. Dans le cas des modifications de solutions, il s'agit plus de la forme des modifications que de leur structure. Nous y donnerons plusieurs idées de forme et un choix sera fait à la fin. Celui-ci servira pour notre premier modèle, du moins dans sa première version.

a. La coulée continue.

Pour représenter la coulée continue, il nous faut au minimum la liste des séquences de coulées caractérisées par leur **nombre de coulées** et leur **nuance**. On aurait pu choisir le tonnage plutôt que le nombre de coulées mais comme ces deux valeurs sont liées par un coefficient constant, on a choisi le nombre de coulées qui sera plus pratique. Il vient de suite que l'**heure de début de coulée** doit être aussi présent. En effet, il est bon de pouvoir se situer par rapport au train lors du calcul de l'enfournement à chaud sans devoir parcourir toute la coulée continue en calculant la longueur de chaque séquence. C'est donc ce triplet de valeur qui caractérisera chaque séquence de coulée.

On peut remarquer que l'heure de début de coulée n'est plus intéressante dans le cas où un parcours total de la coulée est indispensable à chaque calcul. On peut calculer la longueur des séquences au fur et à mesure du parcours de la coulée. Néanmoins, ce calcul peut peser lourd au point de vue temps vu qu'un grand nombre de calcul de l'enfournement à chaud sera nécessaire.

b. Le train.

La séquence de montages du train est une donnée du problème. Le train est un ensemble de montages de profils au sein desquels apparaissent les dérivés. Ces montages de profils sont séparés par des pauses (1 ou 2 heures) nécessaires au changement de cylindre. Le type de dérivé n'apparaît pas explicitement dans la description du train mais les dérivés sont caractérisés par leur vitesse de laminage : la productivité.

Pour représenter le train, il nous faut donc le **tonnage**, la **productivité** et la **nuance** de chaque séquence de laminage au train. Le tonnage est la seule unité de mesure possible par rapport au cas de la coulée continue où le nombre de coulées et le tonnage convenaient aussi bien. En effet, ici, les séquences de laminage ne

se font pas par bloc de grandeur constante mais on lamine des quantités variables d'acier en fonction des désidérata du client. Nous y adjoignons l'**heure de début de laminage** de chaque séquence dans le même but de repérage quasi direct par rapport à la coulée continue.

Une même remarque que pour la coulée continue, quant à ces heures de début de coulée peut être faite.

c. La définition du voisinage de la méthode du Tabou.

Pour celles-ci, plusieurs idées nous sont venues à l'esprit. Certaines en englobent plusieurs de même type. En tout cas, voici celles qui nous ont parues les plus intéressantes. Une description de celles-ci avec leurs avantages et leurs inconvénients est détaillée ci-dessous. Bien sûr, il a fallu choisir parmi ces idées. Nous avons choisi celle qui présentait le plus d'avantages avec un minimum d'inconvénients. Une des qualités que le voisinage devait, à notre avis, au moins présenter, était la facilité de calcul de la variation de la fonction objective lorsque l'on passe d'une solution à une autre.

1ère idée :

La première idée consiste à choisir une des séquences de coulées et d'en modifier ses caractéristiques. Ce choix de la séquence pouvait ce faire de plusieurs manières :

1°) De manière aléatoire : un générateur de nombre aléatoire nous donne l'emplacement de la séquence que nous voulons modifier.

2°) De gauche à droite : nous fixons une première fois une séquence de coulée (peut-être de manière aléatoire ou sur la première séquence) et nous avançons progressivement dans la liste des séquences d'une séquence de coulée à une autre. Une fois arrivé au bout de la liste des séquences de coulées, nous revenons au début de celle-ci et nous continuons à avancer dans celle-ci.

3°) De gauche à droite et de droite à gauche : nous fixons de la même manière que ci-dessus une séquence de coulée et nous avançons progressivement dans la liste des séquences de coulées d'une séquence à une autre. Quand nous arrivons en fin de liste, nous revenons sur nos pas en reculant progressivement dans la liste des séquences de coulées d'une coulée à une autre. Arrivé au début de la liste, nous changeons à nouveau notre sens de parcours vers la fin de la liste.

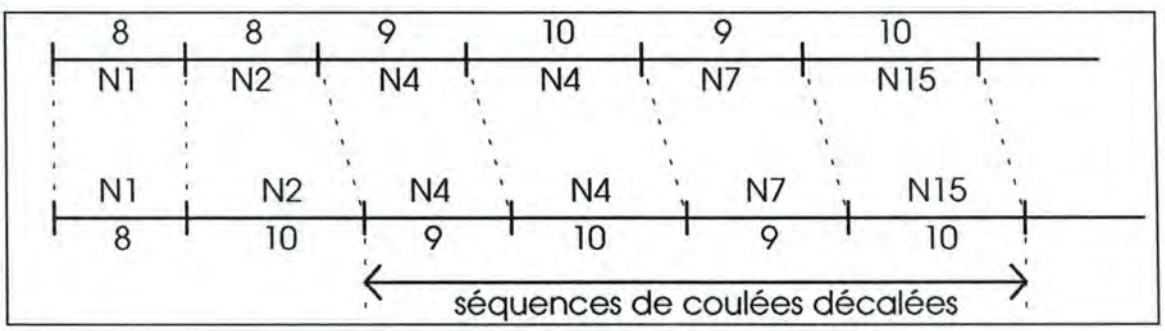
En ce qui concerne les modifications de la séquence, on peut soit changer la nuance seule, soit changer le nombre de coulées.

La modification unique de la nuance de la séquence de coulée choisie, nous permet un calcul aisé de la variation de la fonction objective (qui n'est rien d'autre, dans ce premier modèle, que la valeur du calcul de l'enfournement à chaud). En effet, cette modification n'implique que le calcul de l'enfournement à chaud sur cette séquence de coulée car aucune des autres séquences de coulées n'a été modifiée en quoi que ce soit. Une comparaison de la valeur avant et après modification donne une variation d'enfournement à chaud qui peut être directement ajoutée à la fonction objective pour nous donner sa valeur en cette nouvelle solution. Elle présente aussi l'avantage que, étant donné le nombre limité (fini) de nuances, le voisinage généré n'est pas trop grand et toutes les solutions voisines peuvent être testées.

Par contre, la modification du nombre de coulées sacrifie cette simplicité de calcul de la nouvelle valeur de la fonction objective. En effet, si l'on décroît ou accroît le nombre de coulées, la partie située après cette séquence de coulée est décalée dans le temps et les zones d'enfournement à chaud peuvent varier. On est donc obligé de recalculer l'enfournement à chaud sur toute la partie décalée. Ceci peut prendre beaucoup de temps.

Il faut quand même souligner que le voisinage généré est de cardinal faible. En effet, le nombre de coulées ne peut prendre que 4 valeurs (cas où les bornes min-max du nombre de coulée(s) sont 8 et 11). En combinant celles-ci avec les modifications de nuances, on obtient un voisinage de petite taille et tout le voisinage peut être testé. Ce qui veut dire que l'on peut générer toutes les modifications sur cette séquence de coulée et chercher la meilleure solution de ce voisinage.

Nous avons vu que la variation du nombre de coulées décale toutes les séquences de coulées situées après celle modifiée.



Ceci peut entraîner des pertes de zones d'enfournement à chaud et donc, une diminution de la fonction objective malgré peut-être un gain sur la séquence de coulée modifiée. On peut donc abandonner des solutions qui peuvent présenter

un voisinage intéressant. A l'inverse, nous pouvons gagner de l'enfournement à chaud.

2ème idée :

La deuxième idée consiste à prendre plusieurs séquences de coulées et de les modifier. Ces séquences seraient choisies au hasard dans la liste des séquences de la coulée continue. L'avantage par rapport à la première idée est que la solution construite comporte plus de modifications et qu'un changement plus rapide (vers la solution optimale ou vers des zones non encore explorées) de la solution s'opère.

Les mêmes remarques peuvent être faites ici sur les deux types de modifications. Quand on change uniquement la nuance, le calcul de la nouvelle valeur de la fonction objective se fait aisément mais quand on change la valeur du nombre de coulées, il est nécessaire de calculer à nouveau tout l'enfournement à chaud sur toute la longueur de la coulée continue.

De plus, on assiste à une explosion du cardinal du voisinage. En effet, tous les changements sur une séquence sont à combiner avec tous les changements sur toutes les autres séquences choisies. Quand on choisit plus de 2 séquences, le cardinal de solutions voisines est beaucoup trop important pour qu'on puisse les générer toutes. Un choix doit alors se faire et celui-ci peut s'avérer être le mauvais. Ce qui peut nous conduire vers une solution moins bonne.

3ème idée :

La troisième idée consiste à choisir ces séquences de coulées contiguës c'est-à-dire à choisir un tronçon de quelques séquences. Les modifications sur les nuances se font de la même manière. Mais les modifications du nombre de coulées de ces séquences se font de telle manière que la longueur globale de ces séquences (longueurs additionnées de ces séquences) reste constante. De cette manière, seul l'enfournement à chaud sur ce tronçon doit être calculé. En fait, on supprime le cas de la première idée où il se produisait un décalage d'une partie de la coulée et donc, un calcul entier (sur toute la coulée) de l'enfournement à chaud.

L'avantage de la deuxième solution, à savoir un changement plus rapide de la solution, est présent. Si on ne prend pas plus de deux séquences, le cardinal de solutions voisines n'est pas trop grand et toutes les solutions pourront être testées.

Par contre, on ne pourra essayer toutes les valeurs du nombre de coulées pour chaque séquence car il faut respecter la contrainte de longueur fixe du tronçon. Cet inconvénient peut aller jusqu'à n'avoir aucune possibilité de changement de

valeur du nombre de coulées. En effet, supposons que le tronçon ait une longueur de 2 et que le nombre de coulées soit de 8 pour les deux séquences. La somme des nombres de coulées fait 16. Ce qui fait que la seule combinaison possible est 8 pour la première séquence et 8 pour la seconde. Dans ce cas, les seules solutions du voisinage que l'on peut générer sont celles qui font intervenir un changement de nuance.

Une autre différence qui n'est pas vraiment un inconvénient mais qu'il est bon de souligner, est que dans le cas de la première et de la deuxième idée, on générait exactement le même nombre de solutions dans le voisinage (en supposant que les nuances aient toutes des nombres de coulée(s) min-max identiques). Dans cette troisième idée, cela dépend entièrement des nombres de coulées.

Un autre inconvénient est l'impossibilité de générer certaines solutions d'un type particulier. Ces solutions, bien que spéciales, sont tout à fait possibles et doivent donc être prises au sérieux. La description de ces solutions est la suivante : supposons que les nombres min - max du nombre de coulées soient 8 et 11 pour toutes les nuances présentes sur le train. Une solution où les nombres de coulées de toutes les séquences sont égaux à 8 est impossible à générer. Voyons pourquoi. Supposons que l'on ait une coulée continue de 10 séquences pour lesquelles le total des nombres de coulées est 90. On voudrait que les 7 premières séquences soient à 8. Cela fait 7 x 8 = 56. Et 90-56=34 coulées restantes pour les 3 séquences de coulées. Or 34 / 3 est supérieur à 11 qui est le nombre maximum du nombre de coulées. Il y a donc bien une impossibilité de générer certaines solutions.

Il existe bien sûr un artifice pour pouvoir arriver à toutes ces solutions. Il ne nous restait que 3 séquences de coulées qui devaient recevoir 34 coulées. Si on ajoute une séquence de coulées, nous pouvons alors trouver une solution car il restera alors 34 coulées pour 4 séquences de coulées et 34/4 est inférieur à 11.

Remarque :

Les quantités 8-11 nous ont été données au début de l'étude mais sont susceptibles d'évoluer vers des valeurs plus faibles, par exemple 3-5, qui supprimeront l'impossibilité de générer certaines solutions.

4ème idée :

La quatrième idée n'est en fait qu'un ingrédient supplémentaire pour les deux premières. La modification du nombre de coulées implique le calcul de l'enfournement à chaud sur toute la longueur de la coulée continue tandis que la modification de la nuance n'implique pas ce calcul mais seulement celui de la

variation d'enfournement à chaud sur les séquences choisies. L'idée vient alors de ne faire ce calcul que de temps en temps. Parfois on fait un changement de la valeur du nombre de coulées et parfois on fait un changement de la nuance des séquences choisies ou parfois les deux. Ce choix pourrait être fait par une variable uniforme sur [0,1], soit R.

Si $0 \leq R < b1$ alors modifier la ou les nuance(s) uniquement

Si $b1 \leq R < b2$ alors modifier le ou les nombre(s) de coulées uniquement

Si $b2 \leq R \leq 1$ alors modifier les deux à la fois.

Ces bornes b1 et b2 seraient des paramètres qui permettraient d'orienter le chemin de recherche. On peut en effet les fixer pour tout un problème ou les faire varier pendant ce problème suivant des informations recueillies au début de l'exécution.

Pour la deuxième idée, avec deux séquence choisies, on pourrait faire les modifications uniques des nuances ou uniques du nombre de coulées ou les deux à la fois, sur soit la première séquence de coulée, soit sur la deuxième séquence de coulée ou soit sur les deux. Tout cela en fonction de critères (même forme que ci-avant) bien définis.

d. Choix de structure de solution.

Malgré ses inconvénients, le choix s'est porté sur la troisième idée qui est la seule a nous présentée une simplicité de passage de la fonction objective d'une solution à une autre. Comme nous estimons qu'un grand nombre de solutions doivent être testées, cet avantage est primordial pour ne pas avoir des temps d'exécution trop grand. Il faut remarquer que ce choix s'est basé sur des estimations faites "sur papier" !

Une description plus mathématique de ces éléments que sont les solutions et les modifications associées est réalisée dans la section suivante. Les éléments du Tabou y sont aussi décrits.

E. Le Tabou appliqué : passage en revue des éléments du Tabou appliqué au problème du modèle 1.

Les éléments du Tabou que nous allons étudier sont les suivants :

- les solutions
- les règles de modifications de celles-ci
- la fonction d'évaluation
- les conditions tabous
- les fonctions d'aspiration
- le nombre de détériorations successives avant de s'arrêter

a. Les solutions.

S = ensemble des solutions

$= \{ (c_i , N_i)_{1 \leq i \leq n} \}$

tel que $\forall c_i : c_i \in [8,11]$

et $\forall N_i : N_i \in$ [1,Nombre maximum de nuances]

La longueur "n" est telle que le temps de production de la coulée continue soit supérieur au temps de production du train.

Exemple : [coulée continue]

((10,4) ; (11,9) ; (9,2) ; (8,9) ; (10,4) ; (10,2) ; (10,9) ; (9,2) ; (9,9) ; (9,8))

Nombre de coulées c_i : 10 , 11 , 9 , 8 , ...

Nunace N_i : 4 , 9 , 2 , 9 , ...

b. Les règles de modification des solutions ou mouvement.

M = { mouvement } = { (s,s') | s,s' ∈ S }

Soit $s = ((c_i,N_i))_{1\leq i\leq n}$ une solution quelconque.

Soit p ∈ [1,n-1] la longueur du tronçon.

Nous allons prendre un sous-ensemble N(s) de l'ensemble des mouvements. Il sera défini comme suit :

N(s) = {s' | ∃ k ∈ [1 , n-p] tel que :

$$\forall\, i \neq k,\ldots,k+p-1 : (c_i,N_i) = (c'_i,N'_i) \qquad (1)$$

$$c_k + c_{k+1} + \ldots + c_{k+p-1} = NC \qquad (2)$$

$$c'_k + c'_{k+1} + \ldots + c'_{k+p-1} = NC' \qquad (3)$$

$$NC = NC' \qquad (4)$$

$$N'_k, N'_{k+1}, \ldots, N'_{k+p-1} \in [1, Nmaxnu] \qquad (5)$$

$$\forall\, i \in [1,n] : c'_i \in [8,11] \qquad (6)\ \}$$

La condition (1) indique que, en dehors du tronçon, rien ne change. Les conditions (2) , (3) et (4) indiquent que le total des nombres de coulées dans le tronçon reste constant. La condition (5) indique que les nuances peuvent changer mais dans les nuances autorisées. Enfin, la condition (6) indique que le nombre de coulées doit rester compris entre les limites d'utilisation du répartiteur.

Remarque :

Les nuances sont numérotées de 1 à Nmaxnu car le nom qu'elles portent n'est pas facile à manipuler. On parlera donc de la nuance 4, de la nuance 7, etc.. La liaison avec la réalité (leurs vrais noms) sera faite éventuellement au travers d'un tableau de correspondance s'il y avait besoin c'est-à-dire dans le cas réel d'implémentation.

Quelques exemples :

1°) Tronçon de longueur 2.

((10,4) ; (8,15)) $\Rightarrow$ ((9,7) ; (9,15))

(10 + 8 = 18 = 9 + 9)

2°) Tronçon de longueur 3.

((9,15) ; (10,5) ; (11,13)) $\Rightarrow$ ((11,7) ; (9,5) ; (10;1))

(9 + 10 + 11 = 30 = 11 + 9 + 10)

Le choix de la longueur du tronçon est de p=1. Toutes les solutions accessibles par un mouvement constituent le voisinage N(s) d'une solution. Comme celui-ci peut être très important même pour un tronçon de longueur 2, nous allons choisir un sous ensemble V* de solutions inclus dans N(s). Celui-ci sera composé des solutions accessibles par un mouvement modifiant les nombres de coulées d'au plus une coulée.

Pour être plus précis :

L'ensemble $V^* \subseteq N(s)$ est l'ensemble des solutions $s' = ((c'_i,N'_i))_{1\leq i\leq n}$ tel que :

$\exists\, k \in [1 , n-m]$ tel que : $\forall\, i \neq k,k+1 : (c_i,N_i) = (c'_i,N'_i)$ (1)

$c_k + c_{k+1} = NC$ (2)

$c'_k + c'_{k+1} = NC'$ (3)

$NC = NC'$ (4)

$N'_k , N'_{k+1} \in [1, Nmaxnu]$ (5)

$\forall\, i \in [1,n] : c'_i \in [8,11]$ (6)

$|c_k - c'_k| \leq 1$ (7)

$|c_{k+1} - c'_{k+1}| \leq 1$ (8)

Les conditions (7) et (8) indiquent que les nombres de coulées de la première séquence et de la deuxième séquence de coulées ne peuvent varier de plus d'une coulée. A remarquer que si l'un ne varie pas, l'autre évidemment ne variera pas non plus pour respecter la contrainte (4).

c. La fonction d'évaluation.

La fonction d'évaluation ou fonction objective en une solution est la fonction qui donne la valeur de l'enfournement à chaud pour cette solution. Comme on a choisi une structure de données et des modifications qui n'obligent le calcul de l'enfournement à chaud que sur le tronçon, cette fonction aura un autre paramètre que la solution elle-même. Celui-ci sera l'emplacement du tronçon et sa longueur. Ainsi, on calculera l'enfournement à chaud sur ce tronçon avant les modifications et ensuite on calculera la fonction sur ce même tronçon après les modifications. On obtiendra alors la variation de l'enfournement à chaud sur le tronçon. La fonction objective se calculera aisément comme la somme de la valeur de la fonction objective de la solution avant modification et la variation d'enfournement à chaud sur le tronçon.

d. Les conditions taboues.

Elles vont être représentées par une liste circulaire. Un élément de la liste Tabou sera un triplet :

(i , vc , N).

où 1°) i est un indice désignant un élément de la solution

2°) vc est une variation du nombre de coulées (+1 , 0 , -1)

3°) N désigne une nuance.

Dans notre cas où le tronçon a une longueur de 2, il y aura 2 conditions taboues : l'une sur la première séquence et l'autre sur la deuxième séquence. Un mouvement sera tabou s'il existe deux triplets (i,vc,N) et (i',vc',N') dans la liste tabou tel que le mouvement tabou voudrait modifier l'élément i de la solution par une variation vc du nombre de coulées avec une nuance N et l'élément i' de la solution par une variation vc' du nombre de coulées avec une nuance N'. Exprimons cela plus formellement.

Soit k l'emplacement du tronçon et soit m un mouvement.

$m = s \rightarrow s'$ où :

1°) $\forall\, i : 1 \leq i \leq n$ et $i \neq k,k+1 : s_i = s_i'$.

2°) $c_k(s') = c_k(s) + \text{variation\_coulée}_1$

3°) $c_{k+1}(s') = c_{k+1}(s) + \text{variation\_coulée}_2$

Le mouvement m sera **tabou** si :

$\exists$ t,t' $\in$ T où T est la liste tabou tels que :

1) t = (i,vc,N) et t' = (i',vc',N')

2) i = k et i' = k+1

3) N_k (s') = N et N_{k+1} (s') = N'

4) variation_coulée$_1$ = vc et variation_coulée$_2$ = vc'

La liste des conditions tabous aura une taille proportionnelle à la taille du tronçon. Il est difficile à priori de mettre une valeur à cette taille. Cependant, on peut remarquer qu'avec une taille trop petite la liste tabou ne servirait à rien. En effet, supposons que la liste ait une taille de 2. Il faudrait que dans les deux prochains tronçons, on retombe sur celui qu'on a modifié pour qu'il y ait une "chance" qu'une condition tabou joue. Par contre, avec une taille trop grande, on risquerait de bloquer tout. Supposons à l'extrême que la taille soit de 10 000, il faudrait attendre 10 000 itérations de l'algorithme pour pouvoir revenir à un tronçon semblable. (Il peut être intéressant de revenir à un même tronçon car il se peut qu'après modification du reste de la coulée, on s'aperçoive que c'était ces valeurs qui étaient les meilleures et donc on souhaiterait y revenir) Donc, le choix de cette taille doit se faire par expérimentation. Comme il faut bien mettre une valeur de départ, nous proposons une valeur $\pm$ égale à la longueur de la coulée continue.

e. La ou les fonction(s) d'aspiration.

Nous choisirons pour ce premier modèle une fonction d'aspiration simple : la valeur de la fonction objective pour la meilleure solution trouvée jusqu'alors. Nous dirons que l'aspect tabou d'une modification vers une certaine solution peut être levé si la valeur de la fonction objective en cette solution est meilleure que la valeur de la fonction d'aspiration. C'est en fait un seuil d'aspiration.

La mise à jour de ce niveau d'aspiration, puisqu'ici il s'agit d'une valeur, se fera chaque fois que l'on trouvera une meilleure solution que la meilleure jamais trouvée.

f. Le nombre de détériorations successives avant de s'arrêter.

Cette valeur est le *kmax* dans l'algorithme de la méthode du tabou. Celle-ci est difficile à déterminer à l'avance. Nous fixerons kmax à 1000 mais cette valeur sera très probablement modifiée avec l'expérimentation de notre premier algorithme.

F. Chronologie du développement.

Un des buts du mémoire était de mettre en évidence la démarche de développement d'une méthode et de montrer l'intérêt de développer plusieurs stratégies d'optimisation. C'est pourquoi vous trouverez un détail complet de toutes les étapes de construction, de la version de base décrite dans le paragraphe précédent jusqu'à la version finale de notre modèle que vous trouverez détaillée après ce développement.

Nous allons partir des idées de base qui nous ont permis de construire un premier modèle et voir l'évolution du modèle en fonction de nouveaux éléments ou de nouvelles idées. Ceci n'est qu'une description du développement : une analyse plus profonde sera faite à la section suivante.

L'idée de base est de modifier localement la solution et d'observer les variations de la fonction objective qui en découlent. On choisit donc un tronçon (la longueur du tronçon a été choisie à 2) dans la solution et ensuite, on modifie de telle manière que la longueur du tronçon soit maintenue constante. On a pris soin auparavant de calculer l'enfournement à chaud sur celui-ci. En ajoutant la variation d'enfournement à chaud à la fonction objective, on obtient la valeur de la fonction objective en cette nouvelle solution (solution obtenue par modification de la solution première). Cette valeur peut être alors utilisée pour décider de la qualité de celle-ci.

En ce qui concerne les éléments tabous, le caractère tabou ne porte pas sur la solution elle-même mais sur une modification, celle du tronçon. La modification est tabou si elle vise à mettre dans la première partie et dans la deuxième partie, une situation présente dans la liste tabou. La fonction d'aspiration est en fait un seuil, une valeur qui est celle de la fonction objective prise en la meilleure solution trouvée jusqu'alors (cette solution est notée s* dans l'algorithme du Tabou).

Un nouvel élément entra en jeu : les pauses entre les montages laminés au train. Celles-ci ne sont rien d'autre que des moments où le train ne lamine pas. L'idée est de caractériser ces moments en des zones pour lesquelles l'enfournement à chaud n'est pas possible. On a alors mis des nuances fictives pour ces endroits au niveau de la production du train. Quand l'algorithme veut calculer l'enfournement à chaud, il compare les nuances. Comme celles-ci sont fictives, elles ne correspondent avec aucune nuance de la coulée continue et aucun enfournement à chaud n'est pris en compte. Dans la solution, les modifications des valeurs des nuances ne peuvent prendre ces valeurs de nuances fictives.

Quand on calcule les heures de début de laminage au train, on a souvent des temps inférieurs à l'heure. Comme le type de données pour les heures de début

de laminage sont des entiers, tous les tonnages dont les temps de laminage sont inférieurs à l'heure n'étaient pas considérés. En effet, le résultat de la division du tonnage par la productivité donne un nombre inférieur à 1 et donc, par le troncage résultant du type entier, le temps de laminage est nul. Il fallut donc passer aux réels qui tiendraient compte des décimales d'heure. Ce passage s'avéra un échec car les temps de calcul (addition, multiplication, division et soustraction) prenaient beaucoup plus de temps en type réel qu'en type entier. D'où il résulta des temps d'exécution bien plus important. On travailla alors en entier où la valeur 1 des entiers ne représente plus une heure mais un centième d'heure. On obtint donc une plus grande précision sans perte de temps ou presque (vu les nombres que l'on devait traiter, on dut passer d'entiers en simple longueur (maximum = 32767) à des entiers à double longueur (maximum = 2147483647). Le temps de calcul n'en est que légèrement supérieur. Bien moins, en tous cas, que le passage en réels.

Les modifications sur le tronçon visaient notamment à changer le nombre de coulées. Mais une restriction du problème contraint ce nombre à rester entre deux bornes. Au départ, ces bornes étaient fixes et les mêmes pour toutes les nuances. Puis, on a considéré le fait que celles-ci sont variables en fonction de la nuance mais fixes pour une même nuance durant tout le problème. Ce nouvel élément posa un problème par rapport à notre idée de base de modification locale, à savoir le fait de maintenir la longueur du tronçon constante. En effet, supposons que l'on ait la situation suivante :

Premier élément du tronçon : Nuance N1 avec bornes (8,11)

Deuxième élément du tronçon : Nuance N2 avec bornes (8,11)

Une modification veut les changer en N3 et N4 avec des bornes respectives de (4,5) et (7,8). La longueur minimum du tronçon avant modification est de 8+8=16 tandis que la longueur maximum du tronçon après modification est de 5+8=13. Il est donc impossible de trouver des valeurs du nombre de coulées telle que la longueur du tronçon reste constante.

Ainsi, l'idée de comparaison de l'enfournement à chaud sur le tronçon avant et après modification dut être abandonnée. Par contre, l'idée de génération des solutions voisines par modification locale (le tronçon) peut être gardée. On garde donc le principe de faire varier de $\pm$ 1 ou 0 les éléments du tronçon en maintenant la longueur fixe. On compare maintenant l'enfournement à chaud sur toute la coulée d'avant et d'après modification. Quand on remplace une nuance par une autre dans une séquence de coulées, la moyenne de ces deux bornes min - max du nombre de coulées est la valeur du nombre de coulées que l'on met automatiquement pour cette séquence.

Ce passage au calcul de l'enfournement à chaud sur toute la longueur de la coulée augmenta sensiblement le temps d'exécution. Une nouvelle idée très

intéressante nous apparut. Celle-ci part du fait que seules les nuances en face du tronçon sont susceptibles de fournir l'enfournement à chaud. En effet, le principe de calcul de l'enfournement à chaud est de comparer les nuances du train et de la coulée continue. Comme en général, toutes les nuances ne se trouvent pas en face du tronçon, le nombre d'essais devrait être diminué.

Il y a dans notre problème 15 nuances différentes. Donc, 15 * 15 = 225 combinaisons des deux nuances du tronçon. Si l'on trouve 5 nuances en face du tronçon, cela fait 5 * 5 = 25 combinaisons de nuances. Le gain d'essai est très appréciable. En fait, le nombre de nuances en face du tronçon est fonction du nombre de nuances par montage. On observera d'ailleurs des temps de plus en plus élevés au fur et à mesure que le nombre de nuances par montage est élevé.

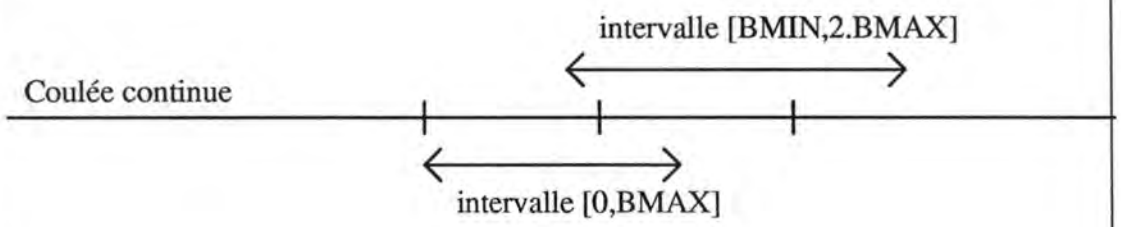
Allons encore plus loin dans cette idée. Essayons sur une séquence de coulées uniquement les nuances se trouvant en face de celle-ci. En effet, par un même raisonnement que ci-dessus, seules les nuances présentes sur le train en face d'une séquence de coulées sont susceptibles de donner de l'enfournement à chaud. Donc, on essayera les nuances situées en face de la première séquence de coulée du tronçon pour la modification de cette séquence et les nuances situées en face de la deuxième séquence de coulées du tronçon pour la modification de cette dernière.

Puisque l'on est amené à recalculer l'enfournement à chaud pour toute la coulée, l'intérêt de garder la longueur du tronçon fixe a disparu. On décidât de passer alors à des modifications locales (le tronçon) mais sans nécessairement garder la longueur du tronçon fixe. On peut donc mettre n'importe quelle valeur de nombre de coulées compris entre les bornes min - max du nombre de coulées associés à la nuance de la séquence de coulées. On calcule l'enfournement à chaud avant et après modification.

L'idée de prendre les nuances en face d'une séquence de coulées pour les essayer dans les modifications de cette dernière bien qu'intéressante, pose un problème. En effet, avant de générer les solutions voisines en modifiant le tronçon, on observait les nuances présentes en face de la première séquence de coulées et ensuite en face de la deuxième séquence de coulées. Il faut remarquer que ces séquences de coulées sont caractérisées entre autre par un nombre de coulées. Mais celui-ci définit la longueur du tronçon, un certain nombre de nuances pouvaient donc manquer. Parfois, celles-ci ne perturbent en rien le déroulement du programme parfois il y a un manque. En effet, supposons que l'on ait au départ (solution à modifier) deux nuances qui ont des bornes (4,5) (nombre min - max du nombre de coulées) et que la première séquence ait 4 comme nombre de coulées et la deuxième, par exemple, 5. On observera les nuances pour la première séquence sur une longueur de 4 coulées et les nuances pour la deuxième séquence sur une longueur de 5 coulées. Supposons qu'il existe des nuances en face, sur le train qui aient des bornes du nombre de coulées de (8,11). L'espace d'enfournement à chaud possible est bien plus grand. On

gâchera donc nos tests en omettant de prendre certaines nuances. On arrive alors à un cercle vicieux. En effet, pour connaître les nuances en face, il nous fait connaître le nombre de coulées. Et pour connaître le nombre de coulées, il nous faut connaître les nuances.

La seule solution possible est la suivante. Il nous faut trouver la plus grande borne possible du nombre de coulées maximum des nuances, soit BMAX cette plus grande borne. On calculera par analogie la valeur BMIN. Ensuite, on observera les nuances présentes en face sur le train sur une longueur de BMAX coulées. Et pour la deuxième séquence de coulées, on observera les nuances sur un intervalle BMIN jusque BMAX multiplié par 2. Nous allons compléter l'explication par un exemple.



Exemple :

Nombre de coulées de la première séquence : 9
Nombre de coulées de la deuxième séquence : 10
Borne minimum BMIN : 8
Borne maximum BMAX : 11

Intervalle [0,BMAX] : [0,11]
Intervalle [BMIN,2.BMAX] : [8,2.11] = [8,22]

Ce problème est nettement plus simple lorsque l'on passe à une seule séquence de coulées comme tronçon. Il suffit de prendre comme intervalle de recherche des nuances en face de notre seule séquence de coulées celui qui commence à 0 coulée et finit à BMAX coulées. C'est-à-dire l'intervalle de temps commençant au début de la séquence de coulées et finissant BMAX coulées après.

Le nombre de solutions voisines est aussi nettement réduit : il n'y a plus que les essais sur une séquence de coulées. Avant, il y avait les essais de l'une conjugués avec les essais de l'autre dans le cas du tronçon de longueur 2. Malheureusement, comme le nombre de solutions voisines est nettement réduit, on peut s'attendre à un changement moins rapide de la solution. Le principe de génération est de toute façon le même mais réduit à une seule séquence.

Il nous parut intéressant d'essayer une dernière modification à notre modèle. Nous allons faire varier la longueur du tronçon. Nous prendrons des tronçons de

longueur 2 et de longueur 1. Une longueur de tronçon supérieure à 2 nous induirait un voisinage beaucoup trop grand. Il faut bien sûr préciser que les modifications de longueur de tronçons se font d'une itération à une autre. En effet, si nous changions cette valeur durant une itération donc durant la génération du voisinage, les solutions de tronçon de longueur 1 modifiées ne seraient pas vraiment comparables à celles de longueur 2. En fait, voici comment se déroule l'algorithme. Tout se passe comme avant sauf au moment de l'appel de la routine qui nous génère notre voisinage. Cet appel est remplacé par un test qui nous envoie soit sur la routine qui génère des tronçons de longueur 1 et soit sur celle qui nous génère des tronçons de longueur 2. Le test est en fait l'appel à un générateur de nombre aléatoire qui pour certaines valeurs nous envoient sur l'une et pour d'autres sur l'autre routine. On a choisi d'orienter légèrement plus (60% des cas) le nombre de génération de voisinage de longueur 2 car l'algorithme avec exclusivement des tronçons de longueurs 2 est un peu meilleur que celui avec des tronçons de longueur 1 exclusivement.

Remarque :

Au cours de ce développement, nous avons accentué les recherches sur le voisinage. Nous devons signaler qu'une modification des conditions taboues s'est opérée au cours de celui-ci. En effet, nous sommes passés d'un algorithme où les changements de nombre de coulées se faisaient par variation de + 1 ou - 1 à un algorithme où l'on pouvait passer d'un nombre de coulées quelconque à un autre nombre de coulées quelconque dans les limites fixées par la nuance. Le troisième élément du triplet qui tenait compte du nombre de coulées (une variation) devait donc être remplacé car il n'avait plus de sens dans notre nouvel algorithme. Nous l'avons remplacer par le nombre de coulées tout simplement. Notre triplet se compose alors de l'endroit de la modification, de la nuance et du nombre de coulées.

Après le paragraphe consacré aux résultats du développement, nous consacrerons une autre paragraphe aux conditions taboues, dans lequel deux types de conditions taboues seront testés.

G. Résultats et analyse de ce développement.

Nous allons procéder à une analyse des résultats du modèle 1. Ils sont le fruit du développement de notre modèle 1. Il est bon de remarquer que tous les tests n'ont pas été effectués sur chaque version du modèle 1, soit pour des raisons de temps, soit parce que les problèmes n'existaient pas encore. En effet, les 6 premiers tests ont été construits de suite pour tester notre modèle tandis que les autres sont venus par après, jugeant qu'il serait bon de les introduire pour mieux tester celui-ci. L'analyse se fera à partir des résultats numériques (enfournement à chaud, temps, ...) et des résultats graphiques (courbes de la fonction objective).

Le tableau suivant montre les caractéristiques des tests. Les numéros les identifient les uns des autres. Le nombre de nuance(s) par montage est le nombre de nuance(s) différente(s) maximum que le test contient par montage. "Nbre dérivés/montage" indique quant à lui, le nombre de dérivés par montage. Le nombre de jour est la longueur du test point de vue du temps. En général, l'optimisation sera hebdomadaire mais on testera le modèle sur des périodes plus grandes. La colonne "productivité" donne une fourchette de productivité dans laquelle les productivités des dérivés ont été choisies.

Test	Nbre nuances par montage	Nbre dérivés par montage	Nbre jours	Productivité
1	2 - 3	3	5	100 - 200
2	2 - 3	3	5	150 - 250
3	1 - 2	3	5	100 - 200
4	1 - 2	3	5	150 - 250
5	4 - 5	3	5	100 - 200
6	4 - 5	3	5	150 - 250
7	10 - 15	3	5	100 - 200
8	10 - 15	3	5	150 - 250
9	2 - 3	3	10	100 - 200
10	2 - 3	3	10	150 - 250
11	1 - 2	3	10	100 - 200
12	1 - 2	3	10	150 - 250
13	4 - 5	3	10	100 - 200
14	4 - 5	3	10	150 - 250
15	10 - 15	3	10	100 - 200
16	10 - 15	3	10	150 - 250

Les 16 tests décrits sont identifiés par leur numéro et leurs caractéristiques pourront être retrouvées dans ce tableau. Les tests 1 à 8 sont les plus réalistes car

l'optimisation est hebdomadaire. Les tests 7 et 8 sont assez rares dans la production.

Pour tous les tests, le nombre de coulées minimum et le nombre de coulées maximum qui sont fonction de la nuance, sont respectivement 8 et 11 pour toutes les nuances.

a. Résultats numériques.

Version 1.

Cette première version de notre modèle comprend l'idée de base (voir "Chronologie du développement") ainsi que la présence des pauses et le passage en précision "centièmes d'heure" tandis que le fait d'avoir des bornes du nombre de coulées min-max en fonction de la nuance n'existe pas encore dans cette version.

L'influence de kmax sur les résultats est inconnu pour notre première implémentation. C'est pourquoi il y a présence d'une colonne "kmax" avec deux valeurs de celui-ci. Le résultat est bien entendu le résultat de notre algorithme, à savoir la valeur de l'enfournement à chaud exprimée en tonne(s). Le nombre d'itérations est le nombre de fois que l'on exécute la boucle principale de notre algorithme. Le temps total d'exécution est exprimé en seconde.

Numéro	Résultat	Kmax	Nbre itérations	Temps total(sec)
1	5652	1000	1506	398
1	5207	200	222	129
2	5464	1000	1422	316
2	4924	200	223	59
3	8270	1000	1421	369
3	6776	200	345	102
4	7971	1000	1023	221
4	7971	200	223	59
5	4524	1000	1023	343
5	4524	200	223	90
6	4758	1000	1024	298
6	4758	200	224	85

Nous pouvons déjà constater l'influence de kmax sur le résultat. Pour rappel, le paramètre kmax de notre algorithme du Tabou est le nombre maximum

d'itérations successives sans amélioration de la meilleure valeur trouvée (notre solution "s-étoile"). Celle-ci se marque assez fort sur les trois premiers tests ; les trois derniers ne donnant, quant à eux, aucune différence sur la valeur du résultat. Cette différence est presque 20% sur le troisième test. On peut déjà dire que, dans l'état actuel de notre modèle, une valeur de kmax mise à 1000 nous paraît être indispensable. Les temps sont alors à multiplier plus ou moins par trois.

Le nombre d'itérations nous donne aussi des informations sur le déroulement de notre algorithme. Nous pouvons remarquer que pour les 3 premiers tests, les résultats avec un kmax=200 et un kmax=1000, sont nettement différents (jusqu'à 20 % de différence de résultat d'enfournement à chaud). Il nous faut donc prendre une valeur de kmax=1000 pour ne pas risquer de passer à côté de solutions intéressantes.

Enfin, l'observation des valeurs des résultats nous fait remarquer des valeurs d'enfournement à chaud de plus en plus grand au fur et à mesure que le nombre de nuances par montage décroît. Pour expliquer cela, il nous faut observer la structure d'un montage. Celui-ci se subdivise en un ensemble d'intervalles d'autant plus nombreux qu'il y a de nuances par montages. Donc, pour un nombre de nuances par montage élevé, nous aurons de petits intervalles. Or, c'est sur ces intervalles que nous observons l'enfournement à chaud ou à froid. La quantité d'enfournement à chaud sera faible pour un petit intervalle et la quantité total d'enfournement à chaud pour toute la coulée continue sera donc plus faible aussi. Bien sûr, la quantité total d'enfournement à chaud dépend de la particularité des valeurs du problème.

Les tests 1 et 2, 3 et 4 , 5 et 6 se différencient l'un de l'autre par leurs gammes de productivité. On ne peut cependant pas les comparer car si leurs caractéristiques nous montrent cette seule différence, la structure (montages, pauses, nombre nuances/montage, etc.) n'est pas nécessairement la même.

Version 2.

Le nombre de coulées min-max des nuances est maintenant introduit et n'importe quelles bornes sont acceptables. Le principe qui vise à prendre uniquement les nuances en face de la séquence de coulée que l'on modifie, est introduit. Le tableau suivant montre les nouveaux résultats liés à cette deuxième version.

Test	Résultat	Kmax	Nbre d'itérations	Temps total (sec)
1	6356	1000	1593	138
1	6114	200	212	23
2	5755	1000	2087	153
2	5451	200	273	27
3	9073	1000	2146	113
3	8413	200	473	26
4	9022	1000	2005	109
4	8133	200	298	18
5	4755	1000	2451	292
5	4467	200	409	53
6	4755	1000	1170	157
6	4755	200	370	53
7	2666	1000	1209	571
7	2666	200	409	202
8	2271	1000	1024	580
8	2271	200	224	136
9	9233	1000	1009	167
9	9233	200	209	37
10	11397	1000	1991	285
10	11212	200	543	79
11	17420	1000	1403	111
11	16607	200	295	27
12	17350	1000	1386	118
12	17350	200	586	53
13	8866	1000	1932	509
13	8373	200	242	76
14	9338	1000	1942	480
14	8848	200	255	72
15	5346	1000	1509	1640
15	5071	200	442	520
16	4684	1000	1286	1314
16	4684	200	486	537

Dix nouveaux tests sont présents. Il s'agit des tests 7, 8,... et 16. Les tests 7 et 8 se font toujours sur une semaine (5 jours) mais avec un nombre de nuances par montage très grand. Ceci, afin de voir le comportement de notre modèle dans des cas extrèmes. Quant aux tests 9 à 16, ils possèdent les mêmes caractéristiques que les tests 1 à 8 respectivement, mais le temps est porté d'une semaine (5 jours) à plus ou moins 2 semaines (10 jours consécutifs). Ce passage à une période plus grande se justifie toujours par le fait que l'on veut observer notre algorithme dans des cas plus difficiles.

La première observation que l'on peut faire, est une nette amélioration de la valeur de l'enfournement à chaud. Celle-ci va de 1% à presque 20%. Il y a juste une exception : le test 6 donne une valeur légèrement inférieure à celle de la précédente version. Cela se fait au détriment du nombre d'itérations qui est nettement plus élevé.

Mais, cette augmentation du nombre d'itérations ne cause pas d'augmentation de temps. Au contraire, une nette diminution du temps d'exécution se remarque. Ce phénomène s'explique tout à fait par la méthode de génération du voisinage. En effet, on ne génère plus un voisinage avec toutes les nuances possibles mais uniquement sur les nuances en face de la séquence que l'on modifie. Ceci fait décroître très fortement le nombre de solutions voisines (sans pour autant en enlever des intéressantes) et donc, on passe moins de temps à tester le voisinage en des solutions inutiles. Le temps consacré à une itération est de ce fait beaucoup plus petit et l'on peut en faire beaucoup plus en moins de temps. Pour montrer la différence de solutions voisines générées, calculons-les. Dans la version précédente, on changeait les nombres de coulées mais pas de la même façon les nuances. Avant, on essayait toutes les nuances c'est-à-dire, comme il y avait 15 nuances (donnée du problème), on faisait 15x15=225 changements de nuance pour un couple de nombres de coulées. Maintenant, cela dépend du nombre de nuances par montage. En effet, plus il y a de nuances par montage, plus la densité de dérivés (et donc de nuances) au sein d'un montage est importante. Par exemple, pour le test 1 et 2, on risque de trouver deux nuances en face de notre tronçon. Pour les tests 7 et 8, ce nombre de nuances peut être supérieure à 10. Il y aura $2.2 = 4$ changements de nuances pour les tests 1 et 2, et, il y aura $10.10 = 100$ changements de nuances pour les tests 7 et 8. Dans un cas comme dans l'autre, le nombre de changements de nuances est nettement inférieur ($4 < 225$ et $100 < 225$). D'où comme le temps d'exécution est dépendant de la taille du voisinage, il diminue fortement pour un même nombre d'itérations. Comme vous pouvez l'obersver dans le tableau précédent, cette deuxième version a besoin d'un plus grand nombre d'itération, mais comme nous gagnions du temps sur une itération (voisinage plus petit), les temps d'exécution sont très inférieurs à la précédente version. Un tableau de comparaison indique l'évolution du temps pour les deux versions de l'algorithme. Les temps donnés sont exprimés en secondes.

Test	1	2	3	4	5	6
Temps Version 1	398	316	369	221	343	298
Temps Version 2	138	153	113	109	292	157
Amélioration	65 %	51 %	69 %	50 %	15 %	47 %

Les raisons du gain de temps de la version 2 par rapport à la version 1 ont été expliquées ci-dessus. En moyenne, ce gain est de 49 % avec au pire une amélioration de 15 %.

Il est toujours justifié de prendre une valeur de kmax=1000 dans le cas du test 11, la différence entre les cas kmax=200 et kmax=1000 est de près de 800 tonnes, soit près de 4,6% de différence. Le test 4 nous montre une différence de près de 10%.

Version 3.

On ne garde plus la longueur du tronçon fixe. On a théoriquement plus de solutions voisines et donc, on passera plus de temps pour une itération.

A partir d'ici, tous les tests ont été exécutés sur un 486 DX II 66 Mhz. Il faut donc faire attention quand on compare les versions 1 et 2 (exécutées sur un 386 DX 40 Mhz) , de diviser les temps de ces deux versions par 3 ou 4 (c'est plus ou moins le rapport de vitesse entre les deux machines)

Test	Temps	Nbre Iter.	Version 3	Version 2	Comparaison
1	20	221	7496	6356	15 %
2	21	276	6313	5755	8,8 %
3	13	223	10198	9073	11 %
4	16	273	9030	9022	0 %
5	32	249	5224	4755	8,9 %
6	26	223	5254	4755	9,4 %
7	132	257	3068	2666	13 %
8	152	284	2851	2271	20 %
9	36	242	14648	9233	36,9 %
10	26	244	12423	11397	8,2 %
11	22	245	20875	17420	16,5 %
12	25	245	18528	17350	6,3 %
13	75	303	10747	8866	17,5 %
14	47	237	10447	9338	10,6 %
15	378	334	6076	5346	12 %
16	348	310	5493	4684	14,7 %

La colonne **version 3** (respectivement **version 2**) est le résultat d'enfournement à chaud pour la version 3 (respectivement 2). La colonne comparaison nous donne le gain en enfournement à chaud que nous procure la version 3 par rapport à la version 2.

On remarque une assez nette progression des résultats d'enfournement à chaud. Cette progression est en moyenne de 12,6%. Cela s'explique par le fait que l'algorithme a plus de liberté de modifications du nombre de coulées. Ainsi, par exemple, dans la version 2, une situation où les deux nombres de coulées du tronçon valaient 8, était bloquée point de vue changement du nombre de coulées. En effet, pour faire un + 1 à l'une des séquences du tronçon, il faut faire un - 1 à l'autre. Or, la contrainte du nombre de coulées minimum empêche de descendre en-dessous de 8. Donc, seuls les changements de nuances seront réalisés dans la génération du voisinage.

Le passage à une valeur Kmax égale à 1000 ne change rien aux résultats d'enfournement à chaud. C'est pourquoi les 16 tests ont été réalisés avec un Kmax égale à 200.

On remarque alors que le temps consacré à une itération est beaucoup plus long dans la version 3. Ceci est tout à fait normal car on génère beaucoup plus de solutions dans le voisinage que dans la version 2 qui, avec toutes les contraintes qu'on lui imposait, ne générait qu'un voisinage de faible cardinal. Malgré cette augmentation du temps consacré à une itération et grâce à la rapidité de la version 3 pour découvrir sa meilleure réponse, la version 3 est nettement positive à tous points de vue par rapport à la précédente.

Version 4.

Cette quatrième version de notre algorithme est celle où l'on rectifie l'intervalle du train sur lequel on prend les nuances destinées aux modifications.

Test	Temps	Nbre Iter.	Résultat
1	25	221	7496
2	21	223	6012
3	16	223	10198
4	19	273	9030
5	45	249	5224
6	38	223	5254
7	187	257	3068
8	203	284	2851
9	49	242	14648
10	43	244	12423
11	25	245	20875
12	29	245	18528
13	114	303	10747
14	74	237	10447
15	512	334	6076
16	482	310	5493

L'observation des résultats du tableau nous montre qu'aucun n'a changé par rapport à la version 3 excepté le test 2. Cette stagnation des résultats est explicable de la manière suivante. Il faut d'abord remarquer que les bornes min-max des nombres de coulées sont 8 et 11 pour TOUTES les nuances. Ensuite, il faut souligner le fait que quand on prenait les nuances en face, nous prenions un intervalle qui était légèrement plus grand que la séquence de coulées. De ce fait, nous considérions déjà les nuances qui se trouve en face d'une séquence plus grande. Nous ne les prenions pas toutes mais le peu que nous prenions, faisait l'affaire.

Version 5.

Cette version 5 nous fait passer d'un tronçon de longueur 2 à un tronçon de longueur 1. Nous pourrons peut-être nous attendre à de moins bons résultats car le voisinage généré sera plus petit et la solution aura plus de difficultés à se modifier. Par contre, une itération devrait prendre beaucoup moins de temps.

Test	Temps	Nbre Iter.	Version 5	Version 4	Comparaison
1	20	1067	7426	7496	- 0,9 %
2	20	1069	6013	6012	0 %
3	19	1115	10259	10198	0,5 %
4	19	1080	9519	9030	5,1 %
5	22	1067	5109	5224	- 2,2 %
6	20	1068	5248	5254	- 0 %
7	39	1052	2857	3068	- 7,3 %
8	41	1063	2535	2851	- 12,4 %
9	30	1366	13976	14648	- 4,8 %
10	26	1181	12324	12423	- 0,8 %
11	24	1139	20737	20875	- 0,6 %
12	25	1118	18528	18528	0 %
13	51	1744	10698	10747	- 0,4 %
14	28	1114	10368	10447	- 0,7 %
15	68	1212	5902	6076	- 2,9 %
16	69	1253	4708	5493	- 16,6 %

La colonne **version 5** (respectivement **version 4**) est le résultat d'enfournement à chaud pour la version 5 (respectivement 4). La colonne comparaison nous donne le gain en enfournement à chaud que nous procure la version 5 par rapport à la version 4.

On remarque de suite une chute des résultats d'enfournement à chaud. Nous l'avions un peu prédit à cause du peu de changement que l'on sait faire d'une solution à une autre.

Comme le temps consacré à une itération est beaucoup plus faible, on s'est permis de mettre Kmax à 1000. Malgré le nombre élevé d'itérations (toujours supérieur à 1050), nous avons des temps très faibles même pour les tests très difficiles comme les tests 7, 8, 15 et 16. Pour ceux-ci, le temps n'est pas supérieur à 70 secondes.

Version 6.

La version 6 est en fait un mélange des deux dernières versions. On a voulu allier la qualité des résultats de la version 4 avec la rapidité d'exécution de la version 5. On prend donc des longueurs de tronçons égales à 1 séquence de coulées ou à 2 séquences de coulées.

Test	Temps	Nbre Iter.	Version 6	Version 4	Comparaison
1	19	252	7496	7496	0 %
2	18	293	6012	6012	7 %
3	12	240	10198	10198	0,6 %
4	13	250	9030	9030	2 %
5	31	309	5224	5224	0,3 %
6	26	261	5254	5254	0 %
7	85	256	3068	3068	0 %
8	93	272	2851	2851	- 2,9 %
9	42	353	14648	14648	- 2,7 %
10	45	454	12423	12423	- 0,8 %
11	24	366	20875	20875	2 %
12	24	309	18528	18528	0 %
13	57	296	10747	10747	- 1,4 %
14	43	275	10447	10447	2,3 %
15	418	646	6076	6076	2,1 %
16	241	346	5493	5493	- 16,6 %

La colonne **version 6** (respectivement **version 4**) est le résultat d'enfournement à chaud pour la version 6 (respectivement 4). La colonne comparaison nous donne le gain en enfournement à chaud que nous procure la version 6 par rapport à la version 4.

On remarque que pour les 7 premiers tests, on gagne aussi bien en rapidité qu'en qualité des résultats. Les autres tests sont plus mitigés mais de toute façon, une chute des temps d'exécution se remarque.

En moyenne, le temps consacré à une itération est plus faible que dans la version 4. Ceci est dû au fait que certaines itérations se font beaucoup plus vite : celles qui génèrent un voisinage à partir d'un tronçon de longueur 1. Ce temps par itération reste néanmoins plus grand que celui de la version 5 car on génère plus de la moitié des voisinages avec des tronçons de longueur 2.

CONCLUSION : QUELLE VERSION CHOISIR ?

Lorsque l'on passe en revue les différentes versions, seules la 4 et la 6 restent en concours. Elles allient toutes les deux une bonne vitesse d'exécution et une bonne qualité des résultats.

Nous avons la chance de pouvoir trouver les optimums de tous les tests par un algorithme assez simple et pas trop coûteux en temps, . Nous allons pouvoir comparer les résultats de ces 2 versions avec les optimums réels.

Test	Optimum
1	5658
2	6519
3	10924
4	9519
5	5332
6	5658
7	3188
8	2956
9	14995
10	13009
11	21369
12	19036
13	10888
14	11572
15	6444
16	5863

Avant de procéder à cette comparaison, il nous faut souligner un détail qui a son importance. Il s'agit de l'influence de notre algorithme par rapport à la suite d'emplacements générés par un générateur de nombres aléatoires. Il apparaît très clairement que notre algorithme ne donne pas le même résultat lorsqu'on varie cette suite. Nous allons donc faire cinq essais (par test) correspondant chacun à un générateur différent de nombres aléatoires. Les 2 tableaux suivants nous montrent ces essais. On y a inclus la colonne reprenant le meilleur essai ainsi qu'une comparaison de ce dernier avec l'optimum.

Version 4 :

Test	Essai 1	Essai 2	Essai 3	Essai 4	Essai 5	Maximum	Erreur
1	7496	7496	7496	7496	7496	7496	0,7 %
2	6012	6519	6253	5970	6012	6519	0 %
3	10198	10259	10259	10198	10198	10259	0,6 %
4	8416	9030	9037	9372	9030	9372	1,5 %
5	5289	5212	5289	5224	5224	5289	0,8 %
6	5254	4846	4846	5254	5254	5254	7,1 %
7	2836	3082	2921	3043	3068	3082	3,2 %
8	2697	2689	2767	2923	2851	2923	1,1 %
9	14705	14439	14505	14705	14648	14705	1,9 %
10	12289	12259	12180	12423	12423	12423	4,4 %
11	20875	20925	21256	20875	20875	21256	0,5 %
12	18528	18528	18528	18528	18528	18528	2,6 %
13	10543	10752	10450	10423	10747	10752	1,2 %
14	10552	10130	11015	10552	10447	11015	4,8 %
15	6004	5445	5864	6167	6076	6167	4,3 %
16	5043	4995	4992	5581	5493	5581	4,8 %

Version 6 :

Test	Essai 1	Essai 2	Essai 3	Essai 4	Essai 5	Maximum	Erreur
1	7556	7496	7496	7496	7496	7556	0 %
2	6519	6354	6219	6219	6519	6519	0 %
3	10259	10924	10924	10924	10259	10924	0 %
4	9046	9519	9519	9223	9223	9519	0 %
5	5269	5250	5106	5332	5242	5332	0 %
6	5254	5254	5254	5254	5254	5254	7,1 %
7	3017	3048	3068	3068	3068	3068	3,7 %
8	2689	2812	2784	2835	2769	2835	4 %
9	14352	14485	14270	14381	14261	14485	3,4 %
10	11947	12259	12591	12423	12318	12591	3,2 %
11	20925	20936	20875	20875	21318	21318	0,2 %
12	18528	18528	18528	18528	18528	18528	2,6 %
13	10438	10740	10654	10641	10594	10740	1,3 %
14	10844	10590	10508	10422	10696	10844	6,2 %
15	5497	6019	6167	6010	6208	6208	3,6 %
16	4708	5103	5665	5122	4708	5665	3,3 %

Le tableau suivant nous compare ces deux versions suivant les erreurs des 16 tests face à l'optimum.

Test	Version 4	Version 6
1	0,7 %	0 %
2	0 %	0 %
3	0,6 %	0 %
4	1,5 %	0 %
5	0,8 %	0 %
6	7,1 %	7,1 %
7	3,2 %	3,7 %
8	1,1 %	4 %
9	1,9 %	3,4 %
10	4,4 %	3,2 %
11	0,5 %	0,2 %
12	2,6 %	2,6 %
13	1,2 %	1,3 %
14	4,8 %	6,2 %
15	4,3 %	3,6 %
16	4,8 %	3,3 %
Moyenne :	2,46 %	2,41 %

Nous ne pouvons décider de la meilleure version sur les moyennes des erreurs des tests car elles sont pratiquement identiques. Par contre, une caractéristique importante se dégage de la version 6 : elle donne 5 fois l'optimum pour les tests 1, 2, 3, 4 et 5. Or, ces tests ont des caractéristiques qui les rendent les plus probables dans notre monde industriel. Pour comparaison, l'optimum n'est donné qu'une fois dans la version 4. (Quand on dit que l'optimum est donné dans un test, cela veut dire que dans un des 5 essais, le test donne l'optimum.)

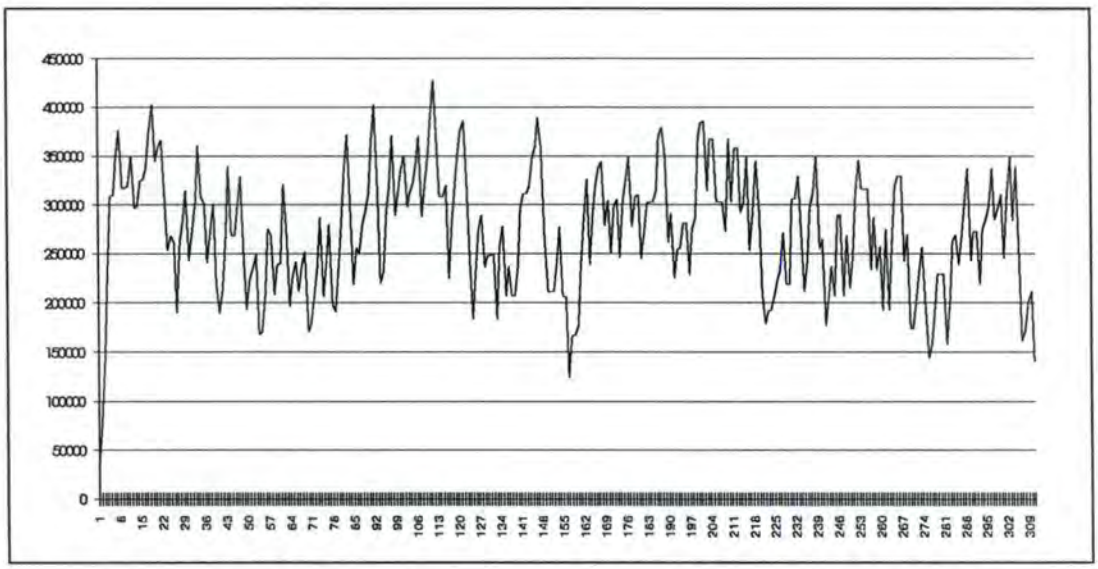
Grâce à cette qualité et à une vitesse d'exécution inférieure à celle de la version 4, notre choix s'est tout naturellement porté sur la version 6. Cette version 6 sera détaillée complètement plus loin pour nous assurer, qu'après cette suite de version d'algorithme, le lecteur sache exactement la structure finale et définitive de notre algorithme.

b. Résultats graphiques.

Les 4 graphiques suivants représentent l'allure de la fonction objective pour un même test (le test 5). Nous allons comparer leurs allures et les expliquer.

Versions 1 et 2.

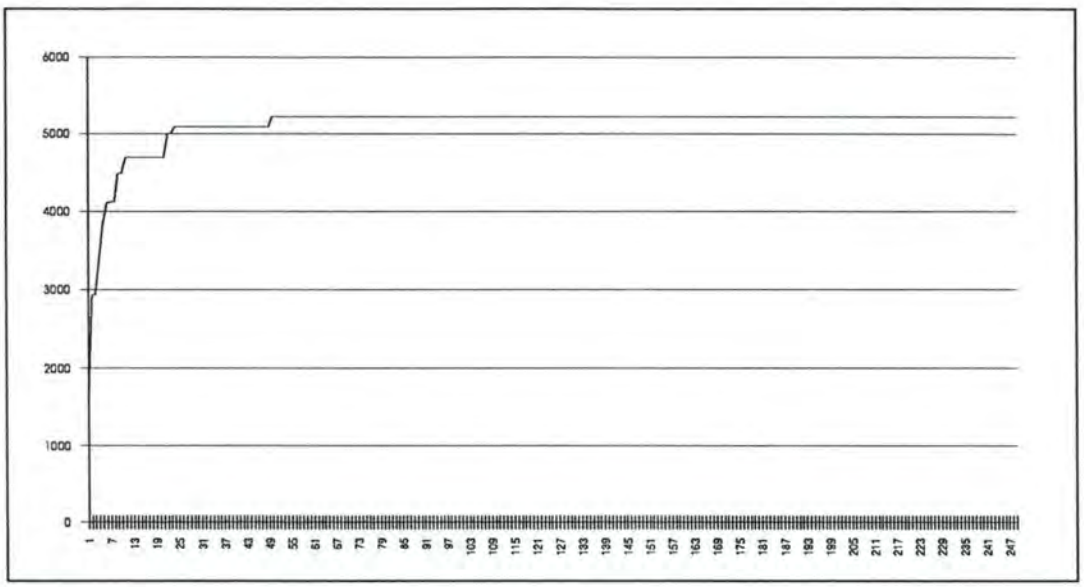
La fonction objective varie très rapidement aussi bien pour croître que pour décroître. Elle se dégrade très fortement à certains endroits.



Ces variations rapides de la fonction objective sont dues à la manière dont on génère notre voisinage dans ces deux premiers modèles. Nous changeons de +1 et -1 nos valeurs de nombre de coulées dans le respect des bornes min-max du nombre de coulées. Il arrivait parfois que l'on ne pouvait pas changer le nombre de coulées (sinon il n'y a avait pas respect des contraintes des bornes). Nous devions alors changer impérativement la ou les nuances pour respecter le principe qui veut qu'on ne reprenne pas la même solution quelle que soit la qualité du voisinage. Et justement, si ce voisinage est de mauvaise qualité, nous avions une dégradation subite de la valeur de l'enfournement à chaud. Ceci explique les variations relativement rapides de la courbe de la fonction objective.

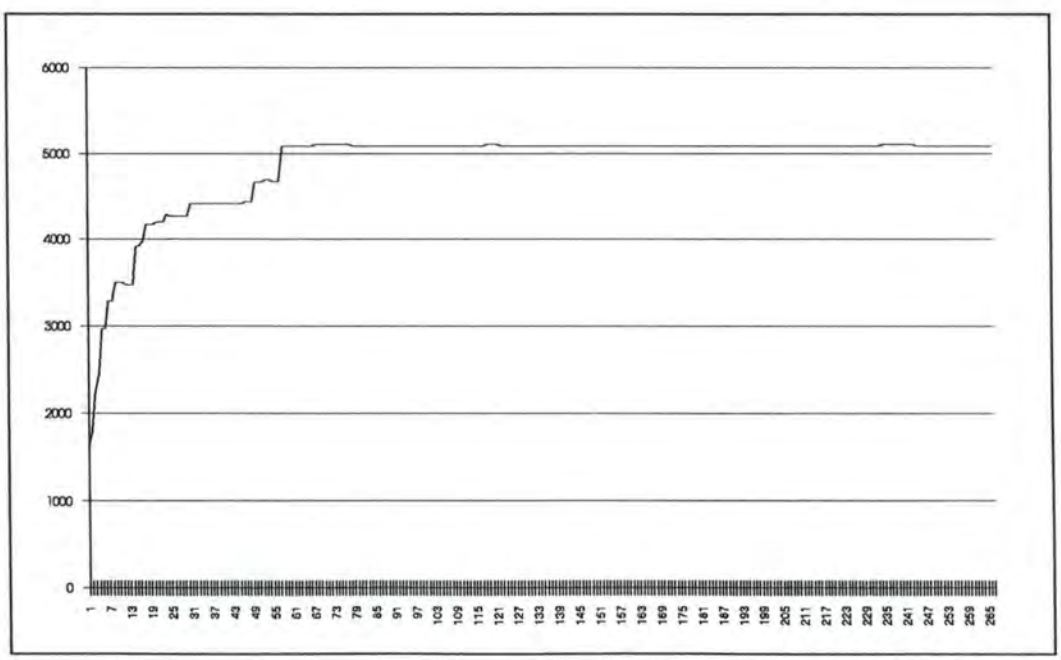
Version 3 et 4.

L'allure de la courbe est nettement différente par rapport aux versions 1 et 2. Il n'y a plus ces variations rapides (diminution ou augmentation) de la valeur de la fonction objective.



La différence avec les deux premiers modèles est que la génération du voisinage n'est plus contrainte (+1 et -1). Il y a une totale liberté de choix du nombre de coulées, dans le respect bien sûr des bornes min-max du nombre de coulées.

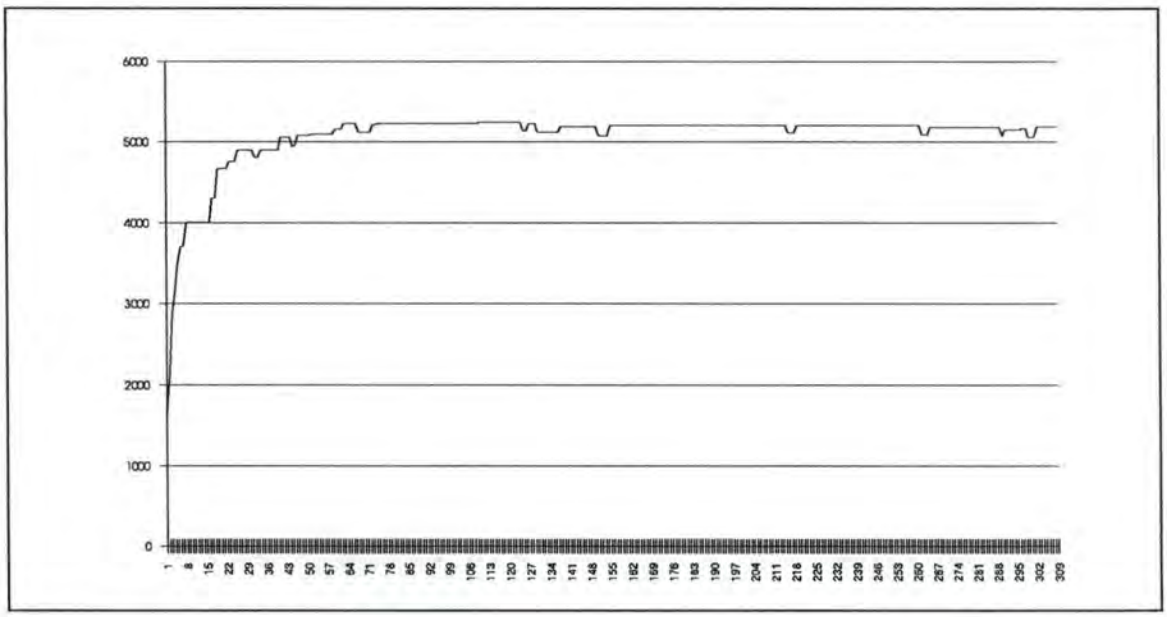
Version 5.



Comme le changement ne s'opère que sur une séquence de coulées dans cette version, il y a une augmentation moins rapide de la fonction objective dans les premières itérations.

Version 6.

Cette version a quelque peu rétabli la forme assez idéale des versions 3 et 4. En effet, les tronçons sont de longueur 2 et de longueur 1. Ce qui permet un changement déjà plus rapide de la solution et donc, une courbe croissant plus rapidement dans les premières itérations.



Remarque :

Les courbes des versions 3,4,5 et 6 sont bonnes du point de vue de la production. En effet, les industriels désirent recevoir de la part de notre modèle, une solution assez rapidement (pour pouvoir avancer dans leur travail). Comme pour ces versions, la courbe croît rapidement dans les premières itérations et peaufine ensuite le résultat, ils obtiennent une solution de qualité assez rapidement.

H. Etude des conditions taboues et de la fonction d'aspiration.

Dans le développement de notre algorithme, nous nous sommes surtout intéressés au voisinage. Nous allons voir maintenant l'influence des conditions taboues, de la longueur de la liste taboue et de la fonction d'aspiration. Nous essayerons aussi de donner une explication.

Pour commencer cette étude, nous allons choisir deux types de conditions taboues pour lesquelles nous ferons quelques tests. Nous essayerons ensuite d'expliquer les résultats associés.

Type 1 :

Un mouvement m sera tabou si :

$\exists$ t,t' $\in$ T où T est la liste Tabou tels que :

1) t = (i,c,N) et t' = (i',c',N')

2) i = k et i' = k+1

3) nuance_k (s') = N et nuance_{k+1} (s') = N'

4) nbre_coulées_k (s') = c et $\text{nbre\_coulées}_{k+1}$ (s') = c'

où s' est la solution sur laquelle on fait la modification.

Type 2 :

Un mouvement m sera tabou si :

$\exists$ t,t' $\in$ T où T est la liste Tabou tels que :

1) t = (i,c,N) et t' = (i',c',N')

2) i = k et i' = k+1

3) nuance_k (s') = N et nuance_{k+1} (s') = N'

où s' est la solution sur laquelle on fait la modification.

Nous pouvons voir que ces deux types de conditions se distinguent par la 4ème condition du premier type. La condition de ce dernier pour que la solution soit taboue est donc beaucoup plus sévère que la condition du deuxième type. Nous devrions donc nous attendre à ce que le jeu des conditions taboues du deuxième type joue plus souvent que le jeu du premier type.

Nous allons pouvoir comparer grâce aux deux tableaux suivants. Le premier tableau donne les résultats avec les conditions taboues du premier type pour le test 1 avec un certain générateur de nombre aléatoire et dans le cas 8-11.

Tableau 1

Lg liste Tabou	Résultats	Nbre iter.	Temps	# sol. taboues	# aspiration
2	7496	252	41/5	0	0
14	7496	252	42/5	1	0
40	7496	300	53/13	9	1
100	7496	353	62/25	30	4
1000	7496	256	97/15	31	3
5000	7496	256	331/53	31	3

Tableau 2

Lg liste Tabou	Résultats	Nbre iter.	Temps	# sol. taboues	# aspiration
2	7496	252	41/5	0	0
14	7496	252	42/5	151	0
40	7496	353	60/23	380	3
100	7496	353	65/25	541	3
1000	7496	463	174/103	1096	3
5000	7496	463	612/375	1096	3

Légende :

- Lg liste Tabou : longueur de la liste Tabou
- Résultats : résultats d'enfournement à chaud
- Nbre iter. : nombre d'itérations du test
- Temps : temps total d'exécution du test / temps pour trouver "l'optimum"
- # sol. taboues : nombre de solutions taboues pour tout le test
- # aspiration : nombre de solutions taboues levées par aspiration

Le première constatation que nous pouvons faire est la constance du résultat quel que soit le type de condition taboue et la longueur de la liste Tabou. Cette dernière influe très nettement sur le temps d'exécution qui est multiplié par plus de 10 lorsque l'on passe d'une longueur de liste de 2 à 5000. Ensuite, nous pouvons observer que le nombre d'itérations pour y arriver est très souvent différent. En ce qui concerne les conditions taboues, le premier type joue mois souvent que le deuxième. Mais l'aspiration joue presqu'autant.

Qu'est ce que cela signifie ? Pour la constance du résultat, il apparaît par d'autres tests que cette spécificité est courante mais pas systématique. En effet, par exemple pour le test 9, il apparaît un résultat différent pour les exécutions avec une longueur de liste comprise entre 2 et 40, et, une longueur de liste comprise entre 100 et 5000. Mais cette différence est assez faible : moins de 2 %.

Comment expliquer que le résultat reste constant alors que le nombre de solutions taboues et le nombre de solutions taboues levées par aspiration n'est pas toujours le même (lorsque l'on change la longueur de la liste) ? Ces nombres sont peut-être différent mais le chemin aussi vu que le nombre d'itérations est différent d'un test de longueur de liste à l'autre. Il apparaît aussi que comme tout les tests ne donnent pas toujours le même résultat lorsque l'on change la longueur de la liste, cette constance est purement fortuite !

L'influence de la longueur de la liste Tabou sur le temps d'exécution s'explique très facilement. Plus la liste est grande, plus test de présence d'un élément dans la liste prend du temps. Nous pouvons constater que pour des listes trop grandes, le temps d'exécution devient trop grand. Nous pouvons donc dire qu'il ne faut pas prendre une liste trop grande au risque de voir le temps d'exécution prendre des valeurs énormes.

En ce qui concerne les différences entre les nombres de modifications taboue et d'aspiration entre les deux types de conditions taboues, nous avons déjà expliquer cela par la sévérité plus grande du premier type par rapport au deuxième.

Il nous reste à parler de la fonction d'aspiration. D'après les résultats, celle-ci ne joue pas beaucoup. Vu que celle qui a été implémentée est assez simple (aspiration aisée) et qu'elle ne joue pratiquement pas, nous ne voyons pas l'intérêt de la modifier pour tester son influence.

I. Passage en revue des éléments de notre algorithme dans sa version finale

a. Les solutions.

$S = (c_i , N_i)_{1 \le i \le n}$

telle que : 1) $\forall\ c_i : c \in [\text{borne\_inf}_{N_i} , \text{borne\_sup}_{N_i}]$

2) $\forall\ N_i : N_i \in [1,\text{Nmaxnu}]$

b. Les modifications.

Les tronçons que l'on va modifier sont de deux tailles différentes : des tronçons d'une séquence de coulées et des tronçons de 2 séquences de coulées. C'est pourquoi on voit apparaître deux formes de voisinage. Ils seront choisis de manière aléatoire en favorisant un peu plus les tronçons à deux séquences de coulées (60 % des cas).

Pour les tronçons de deux séquences de coulées, le voisinage généré est :

$N(S) = \{ S' \mid \exists\ k \in [1,n-1]$ tq :

$\forall\ i \neq k , k+1 : (c_i,N_i) = (c_i',N_i')$ (1)

$N'_k , N'_{k+1} \in$ Nuances_en_face (2)

$c'_k \in [\text{borne\_inf}_{N_k}, \text{borne\_sup}_{N_k}]$ (3)

$c'_{k+1} \in [\text{borne\_inf}_{N_{k+1}}, \text{borne\_sup}_{N_{k+1}}]$ (4) $\}$

Pour des tronçons d'une séquence de coulées, le voisinage généré est :

$N(S) = \{ S' \mid \exists\ k \in [1,n]$ tq :

$\forall\ i \neq k : (c_i,N_i) = (c_i',N_i')$ (1)

$N'_k \in$ Nuances_en_face (2)

$c'_k \in [\text{borne\_inf}_{N_k}, \text{borne\_sup}_{N_k}]$ (3) $\}$

Dans les conditions de génération du voisinage, nous allons expliquer la condition 2 : "$N_k \in$ Nuances_en_face". La nuance N_k doit appartenir à l'ensemble des nuances du train qui se trouvent en face de la séquence de coulée k. Nous allons essayer d'exprimer formellement l'intervalle "en face" et de celui-ci nous retirons les nuances (nuances_en_face).

Soient c et d les heures de début et de fin de la séquence de coulées.

Soient h_i les heures de début des montages au train.

Soient a et b les bornes de l'intervalle "en face".

$a = h_j$ tel que

1) $h_j \leq c$

2) $\forall h_i : h_i \leq a$ et $h_i < h_j$

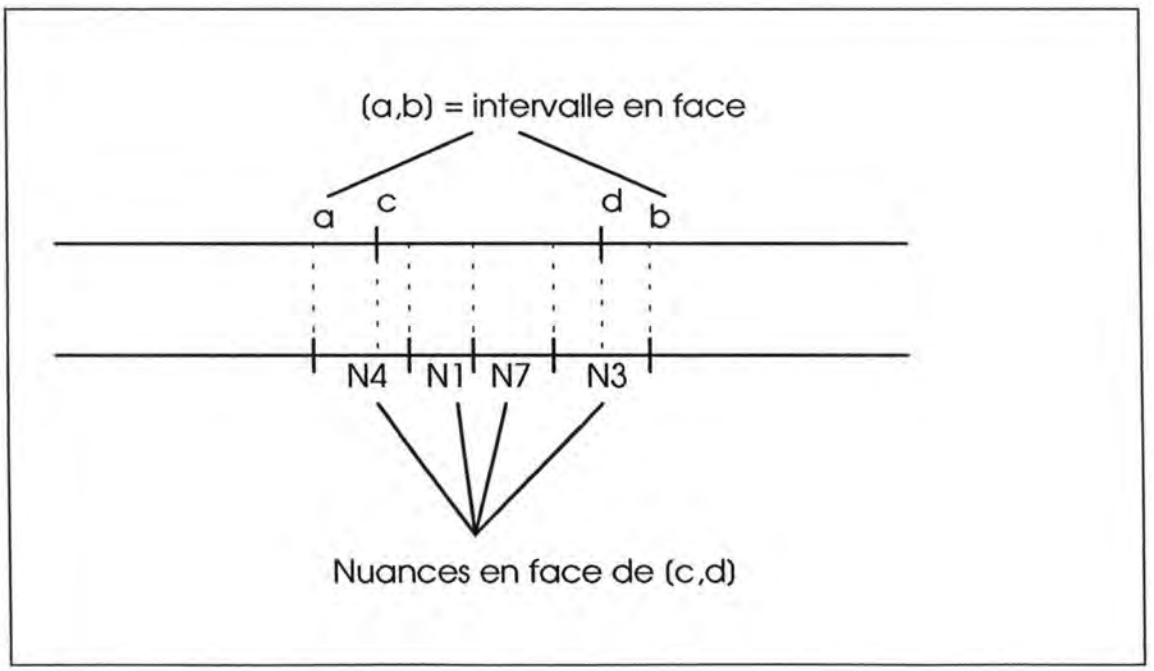
$b = h_p$ tel que

1) $d \leq h_p$

2) $\forall h_q : b \leq h_q$ et $h_p < h_q$

Cet intervalle contient donc une période de production du train, qui contient un certain nombre de tonnages à produire. Ceux-ci sont en outre caractérisés par une nuance. Nous pouvons alors définir l'ensemble N comme l'ensemble des nuances des tonnages présents dans l'intervalle [a,b].

On peut donc définir "nuances_en_face" comme l'ensemble N. Le dessin explicatif va éclairer cette notion de nuances en face. Il faut noter que seuls les nuances "réels" sont considérées; les nuances fictives pour les tonnages fictifs représentants les pauses sont exclues de "Nuances_en_face".



c. Fonction d'évaluation.

Avec la possibilité de mettre n'importe quelles valeurs comme nombre de coulées pour les séquences de coulée, la longueur du tronçon ne reste plus fixe et on est obligé de calculer l'enfournement à chaud sur toute la longueur de la coulée continue à chaque fois que l'on fait une quelconque modification sur notre solution. Dans un souci d'optimisation du temps, on pourrait gagner quelque peu en ne calculant l'enfournement à chaud que sur la partie modifiée c'est-à-dire le tronçon et la partie gauche par rapport au tronçon de la solution. En moyenne, on ne calculerait l'enfournement à chaud que sur la moitié de la longueur de la coulée continue, ce qui est peut être appréciable.

d. Conditions taboues.

Nous avons maintenant un algorithme dont le voisinage est un tronçon de d'une séquence de coulées ou de deux séquences de coulées. Dans le cas d'un tronçon de deux séquences de coulées, nous avons deux conditions taboues : une condition taboue sur la première séquence de coulées et une autre taboue sur la deuxième séquence de coulées. Dans le cas d'un tronçon d'une séquence de coulées, nous avons une condition taboue. Le statut tabou d'une modification s'exprime maintenant comme suit :

1) pour un tronçon d'une séquence de coulées :

Le mouvement sera tabou si :

$\exists\, t \in T$ où T est la liste Tabou tel que :

1) $t = (i,c,N)$

2) $i = k$

3) $\text{nuance}_k(s') = N$

4) $\text{nbre\_coulées}_k(s') = c$

où s' est la solution sur laquelle on fait la modification.

2) pour un tronçon de deux séquences de coulées :

Le mouvement sera tabou si :

$\exists$ t,t' $\in$ T où T est la liste Tabou tels que :

1) t = (i,c,N) et t' = (i',c',N')

2) i = k et i' = k+1

3) nuance_k (s') = N et nuance_{k+1} (s') = N'

4) nbre_coulées_k (s') = c et $\text{nbre\_coulées}_{k+1}$ (s') = c'

où s' est la solution sur laquelle on fait la modification.

La longueur de la liste taboue a été choisie proportionnelle au nombre de séquences de la coulée continue c'est-à-dire pour un test portant sur 5 jours, nous mettrons une liste de longueur 14 ($\approx$ nombre de séquences de la coulée continue).

e. Fonction d'aspiration.

La fonction d'aspiration est la même que dans la section D de ce chapître. C'est donc un seuil dont la hauteur est la valeur de la fonction objective en la meilleure solution trouvée jusqu'alors. On met donc à jour cette fonction à chaque fois que l'on trouve une meilleure solution.

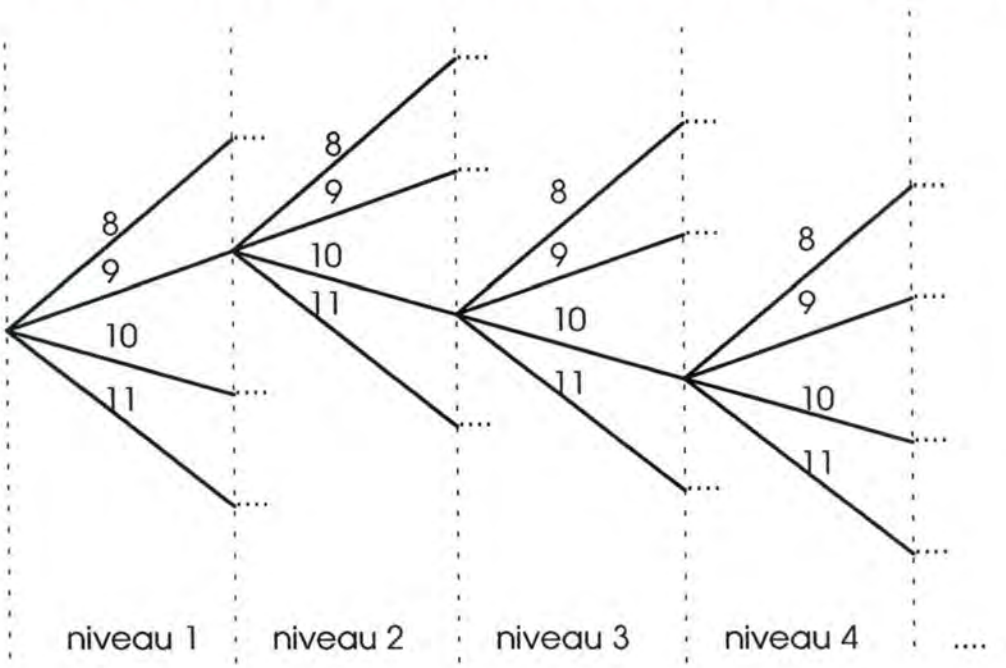
f. Nombre de détériorations successives avant de s'arrêter.

Etant donné que notre algorithme est assez rapide pour faire ces modifications et que donc la meilleure solution est trouvée assez rapidement, on a fixé "kmax" à une valeur pas trop grande. Une valeur de "kmax" de 200 suffit pour s'assurer que l'algorithme trouve sa réponse optimale.

J. Génération des solutions possibles d'un test en vue de trouver l'optimum réel du test.

Nous voulions trouver l'optimum pour les tests que nous avions créés. Pour cela, nous avions dans un premier temps généré toutes les solutions possibles et choisi la meilleure d'entr'elles suivant le critère d'enfournement à chaud. En fait, il s'agit de développer un arbre dont le poids des arêtes est le nombre de séquences de coulées et de trouver la branches de poids maximum. Comme le nombre de branches est très grand lorsqu'on développe celui-ci à des niveaux allant jusquà 15 ou plus, ce procédé prenait beaucoup de temps machine et un temps de plusieurs jours pour certains tests (tests dont la durée est de 10 jours).

ARBRE INDUIT PAR LE NOMBRE DE COULEES DES SEQUENCES



Nous avons alors trouvé une autre méthode beaucoup plus rationnelle : la programmation dynamique. Elle se base sur l'idée de ne pas continuer à développer des branches qui ne sont plus intéressante. Le graphe dont on parle ici, est celui induit par les choix (8 à 11) du nombre de coulées de chaque séquence de coulées, et n'est plus un arbre comme on le verra plus loin. On continuera à appeller branche une suite d'arète du graphe. Le poids des arètes est toujours le nombre de séquences de coulées. Sur ces arètes vient se greffer la

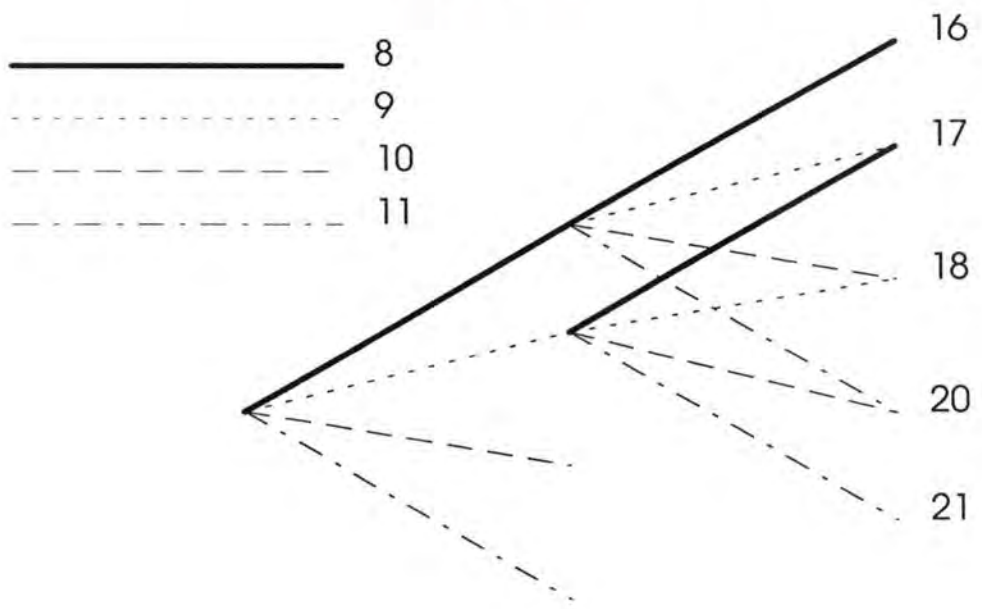
nuance associée à la séquence de coulées (elle n'apparaît pas dans le dessin). Le critère de sélection de ces branches (branches qui ne sont plus intéressantes) est le suivant. Supposons que nous avons développé le graphe jusqu'au niveau n. Supposons ensuite que nous avons deux branches différentes qui se terminent à un même moment (même heure). Voici ces branches en cours de développement.

(9,10,11,8,10, ... à développer ....)

(8,11,11,9,9, ... à développer ....)

Ces deux branches ont été développées jusqu'au niveau 5 (5 séquences de coulées) et se terminent toutes les deux à la 48ème heure. On peut donc décider pour laquelle on va abandonner le développement. En effet, puisqu'elles se terminent à la même heure, la suite ne peut qu'être strictement identique du point de vue de l'apport en enfournement à chaud. En effet, si on développe ces deux branches, on choisira la même fin de branches, uniquement par le fait qu'elle SE TERMINE A LA MEME HEURE. Comme la suite sera identique, le choix entre ces deux graphes (partie développée + graphe restant à développer) se fera uniquement sur la partie déjà développée. On va donc comparer l'enfournement à chaud sur les parties déjà développées de ces deux branches. Celle qui en aura créé le plus sera choisie et l'autre sera abandonnée. Donc, le graphe induit est un arbre pour lequel certaines branches se sont rejointes. C'est donc pourquoi le graphe n'est plus un arbre. Le dessin suivant montre quelques jointures de branches.

GRAPHE INDUIT PAR LE NOMBRE DE COULEES DES SEQUENCES

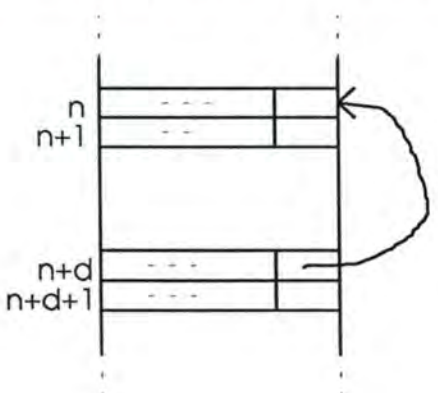


Comment va-t-on faire cela pratiquement ? On a besoin d'un tableau de dimension égale au nombre d'heure du test pour lequel on veut trouver l'optimum. La case n de ce tableau contiendra deux choses. La première est une valeur d'enfournement à chaud. Celle-ci sera l'enfournement à chaud de la meilleure branche développée jusqu'alors. La deuxième chose est un couple permettant de reconstituer la solution par après. Ceci pouvant être facultatif si l'on ne s'intéresse qu'à l'optimum sans vouloir connaître la forme de la solution qui donne cet optimum. Ces deux choses sont une nuance et un pointeur vers une case du tableau à partir de laquelle on a atteint cette case contenant le pointeur. Supposons que se soit les case m et p dont on parle ici, cela signifiera que il y a une séquence qui commence à l'heure m et se termine à l'heure p. Le principe est le suivant : il faut développer la plus petite case qui ne la pas encore été. Supposons que ce soit la nième. On va calculer l'enfournement à chaud maximum que l'on peut obtenir sur l'intervalle de temps [n , n + d] où d est le nombre de coulées (8 à 11).

Ce calcul se fait comme suit : on parcourt cette intervalle sur le train en additionnant pour chaque nuance séparée le tonnage à chaud. A la fin de celui-ci, on choisit la nuance qui donne le plus de tonnage à chaud sur cette intervalle; elle est attribué à cette séquence de coulées caractérisée par l'intervalle. On a donc l'enfournement à chaud sur cette séquence (noté EAC_SEQ).

On va alors comparer la somme de EAC_SEQ avec la valeur de la case n, avec la valeur de la case n+d. Si cette somme est plus grande, cela veut dire qu'on a trouvé une branche se terminant en (n+d) heures et qui donne plus d'enfournement à chaud sur les (n+d) heures que celle qu'on avait trouvé jusqu'alors. Il est évident qu'on choisira de continuer le développement de celle-ci plutôt que de l'ancienne trouvée.

REPRESENTATION DU TABLEAU



Certaines cases du tableau ne sont pas atteintes : ce sont des heures pour lesquelles il est impossible qu'une suite de séquences de coulées se terminent à cette heure (heure désignée par la case). Un artifice (-1) permettra de ne pas développer ces cases qu'il ne faut évidemment pas développer. On s'arrêtera de développer lorsque l'heure désigné par la case que l'on doit développer, est supérieure à l'heure de fin du train.

K. Les ingrédients du modèle 1.

Comme le modèle est très simple, il n'y a guère la possibilité d'avoir des ingrédients. Nous avons trouvé un ingrédient qui est en fait apparu dans notre développement.

Quand il a été question de choisir entre 2 versions du modèle 1, la 4 et la 6, nous avons fait plusieurs essais car notre algorithme ne donnait pas toujours nécessairement la même valeur si nous suivions des chemins différents. Il faut alors remarquer que la solution initiale a aussi son importance.

Un ingrédient possible est de faire un certain nombre d'essais en faisant varier soit la solution initiale, soit le chemin des emplacements de tronçons. Ainsi, nous avons plus de chance de mieux parcourir l'ensemble des solutions.

Le choix de la solution initiale peut se faire de manière aléatoire pour les différents essais. Mais nous pourrions aussi retenir les zones visitées de l'ensemble X des solutions, lors des derniers essais et ainsi, choisir une solution initiale dans une zone peu ou pas du tout fréquentée.

L. Conclusions premières sur la méthode du Tabou.

a. Résultat proprement dit de l'algorithme.

En regardant les résultats comparés à l'optimum, nous pouvons être satisfaits du comportement du Tabou. En effet, il donne 5 fois l'optimum sur 16 tests et il est très bon (- de 2,5 % d'erreur en moyenne) sur les autres tests. Les temps d'exécution sont tout à fait satisfaisants avec moins de 2 minutes pour presque tous les tests. Le 15 et 16ème tests prennent respectivement moins de 7 et 5 minutes. Il s'agit de tests extrêmes.

Malgré ces résultats intéressant, nous pouvons avoir plusieurs regrets. D'une part, l'algorithme est dépendant de la solution initiale et d'autre part, il est aussi dépendant du générateur de nombres aléatoires. En effet, plusieurs essais s'imposent pour être sûr d'obtenir une bonne réponse. Mais comme toujours dans ce genre d'essai, nous pouvons faire 5 essais avec 5 générateurs différents de nombres aléatoires et n'obtenir qu'un résultat moyen alors qu'il en existe un 6ème qui donne une très bonne réponse. Ce choix pour les essais ne fait qu'ajouter un paramètre de plus à notre algorithme.

b. Les conditions taboues dans la génération du voisinage.

Le système des conditions taboues pour éviter les minimums locaux a l'air de fonctionner. Ce doute subsiste car en fait, nous observons bien l'élimination de certaines solutions à visiter mais nous ne sommes pas sûr qu'il le fait bien à propos. Evidemment, il n'est pas très évident de le savoir.

Nous avons vu que ni la longueur de la liste, ni la structure (celles qu'on essayée) des conditions taboues n'influencent notre algorithme mais cela est peut-être du à la particularité de notre problème.

c. La fonction d'aspiration.

Le système de levée du statut tabou par la fonction d'aspiration n'est quand à lui pas très utilisé. Est-ce parce que la fonction d'aspiration est mal choisie ou parce qu'il n'y en a pas besoin ? Encore une fois, il n'est pas évident de le prouver ou même de le voir. Enfin, lorsque la fonction d'aspiration joue, notre algorithme prend un chemin différent (ce qui est normal) mais nous retrouvons le même résultat.

Chapitre III : Le modèle 2

A. Enoncé input - output du problème du modèle 2.

L'énoncé du problème du modèle 2 est le même que celui du modèle 1 auquel on a adjoint une condition sur le stock de beams blanks : étant donné des besoins en acier pour le train, qui sont caractérisés par une nuance, une quantité et une productivité, trouver un ordre et les caractéristiques des séquences de coulées à la coulée continue afin de maximiser l'enfournement à chaud tout en veillant à ce qu'aucun stock de beams blanks d'aucune nuance ne prenne une valeur négative tout au long de la production.

La seule différence entre le modèle précedent et celui-ci est le stock de beams blanks pour chacune des nuances. Dans le modèle 1, ce stock apparaissait de manière infinie pour chacune des nuances. Vu qu'il n'avait aucune influence sur notre optimisation, il n'était pas mis à jour. Dans ce modèle 2, il faut non seulement le mettre à jour mais tenir compte de son influence dans le déroulement de l'optimisation.

Input :

Le stock initial.

Une suite de nuances avec pour chacune d'elles, un stock ***limité*** (niveau en début de production), une borne inférieure et une borne supérieure (ces deux bornes sont les limites d'utilisation du répartiteur pour cette nuance d'acier), c'est-à-dire :

$$(\text{Nuance}_i , \text{stock\_initial\_nuance}_i , \text{borne\_inf}_i , \text{borne\_sup}_i)_{1 \le i \le \text{Nbre\_nuance}}$$

où :

1) Nbre_nuance est le nombre de nuances différentes que la coulée continue peut fournir.

2) $\text{Stock\_initial\_nuance}_i < +\infty , \forall\, i : 1 \le i \le \text{Nbre\_nuance}$.

Le train.

Une séquence de montages pour une semaine avec nuance, tonnage et productivité, c'est-à-dire :

$$\text{Train} = \{ (\text{Tonnage}_i , \text{Productivité}_i , \text{Nuance}_i , \text{Heure de début}_i)\ 1 \le i \le lg\_tr \}$$

où "lg_tr" est le nombre total de tonnages à produire, chaque tonnage correspondant à une nuance et une productivité pour ce tonnage.

Ouput :

La coulée continue.

Les séquences de coulées à la coulée continue avec nuance et nombre de coulées, c'est-à-dire :

$$\{ \text{C.C.} = (\text{Nombre\_coulées}_k , \text{Nuance}_k)\ 1 \le k \le Lg\_cc \}$$

où :

1) Lg_cc est la longueur de la liste des séquences de coulées.

2) $\text{Nombre\_coulées}_k \in [\ \text{Borne\_inf}_{\text{nuance}_k} , \text{Borne\_sup}_{\text{nuance}_k}\ ]$

Le stock final.

Un stock pour chacune des nuances (stock obtenu à partir de "stock_initial_nuance" en fin de production du train et de la coulée continue) c'est-à-dire :

$$\text{Stock} = (\ \text{Stock\_final\_nuance}_k\ )\ 1 \le k \le \text{Nbre\_nuance}$$

où $0 \le \text{stock\_final\_nuance}_k < +\infty$, $\forall\ k : 1 \le k \le \text{Nbre\_nuance}$

B. Exemple numérique du modèle 2.

Dans cet exemple du modèle 2 se trouve essentiellement les tableaux qui ont évolués ou les nouveaux tableaux. En fait, seul le tableau avec la description des nuances (ou stock) a évolué. Sa colonne "stock" ne contient plus des valeurs infinies comme dans le modèle 1 mais bien des valeurs finies représentant le stock initial. On trouve un nouveau tableau : le stock final. Enfin, nous tracerons pour une nuance l'évolution du stock en fonction du temps (du déroulement de la coulée continue et du train).

Le stock initial :

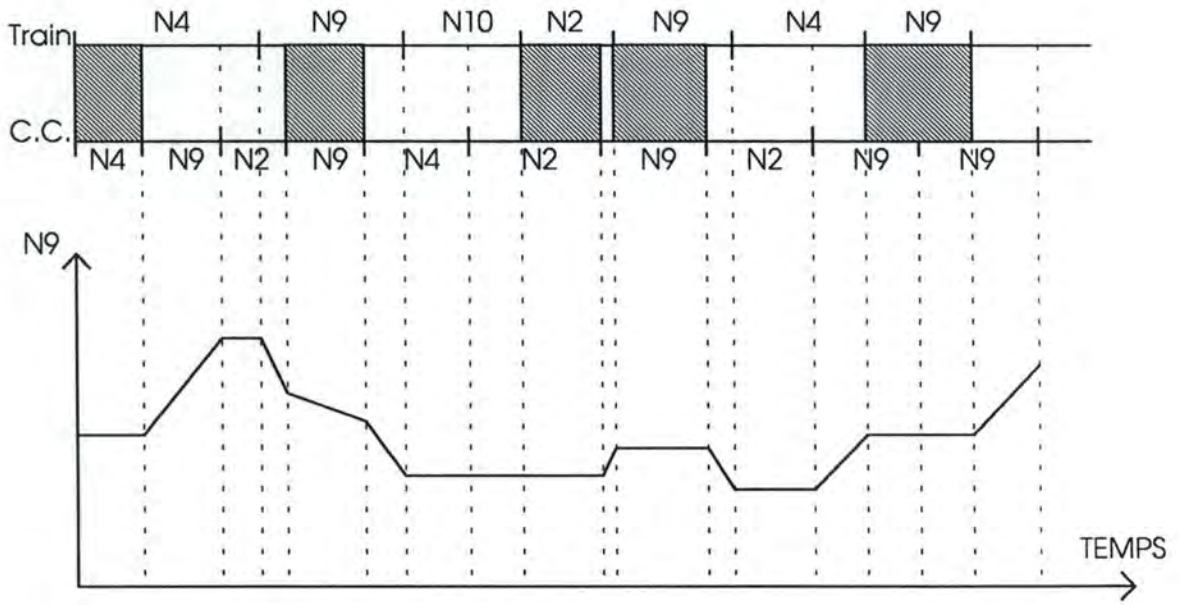
Nuance	Stock	B.I.	B.S.
1	0	3	5
2	628	3	5
3	0	3	5
4	937	3	5
5	245	3	5
6	0	3	5
7	275	3	5
8	189	3	5
9	6	3	5
10	240	3	5
11	88	3	5
12	0	3	5
13	130	3	5
14	141	3	5
15	0	3	5

Le stock final :

Nuance	Stock
1	150
2	138
3	50
4	86
5	1521
6	12
7	189
8	275

9	90
10	250
11	89
12	120
13	10
14	191
15	78

On pourra facilement tracer l'évolution du stock d'une nuance en fonction du temps, s'il y avait besoin. Le dessin suivant montre l'évolution de la nuance 9 pour une coulée continue dont les caractéristques sont ici quelconques.



C. Calcul du stock et de l'enfournement à chaud.

Dans un paragraphe C du chapitre II, nous avons expliqué le principe de production et notamment, les entrées/sorties sur le stock. Nous avions aussi exprimé formellement le calcul de l'enfournement à chaud au fur et à mesure de la production de la coulée continue et du train. A cette fin , nous avions défini : zone d'enfournement à froid, zone d'enfournement à chaud et valeur de l'enfournement à chaud pour une zone d'enfournement à chaud.

De la même manière que nous l'avions fait pour le calcul de l'enfournement à chaud, nous allons exprimer l'évolution du stock au fur et à mesure des zones définies par le train et la coulée continue. Ensuite, après avoir remarquer la symétrie parfaite entre le calcul du stock et le calcul de l'enfournement à chaud, nous fusionnerons ces deux calculs en une même procédure.

Soit $(Z_i)_{1 \le i \le \#zones}$ l'ensemble des zones définies par le train et la coulée continue.

Soit $Z_k \in (Zi)_{1 \le i \le \#zones}$, Z_k est caractérisée par 6 éléments :

1) la nuance NT_k et la productivité PT_k sur le train

2) la nuance NCC_k et la productivité PCC_k sur la coulée continue

3) les début A_k et fin B_k de zones

Nous savons qu'il existe deux types de zones :

1) les zones d'enfournement à froid

2) les zones d'enfournement à chaud

Nous allons exprimer les variations de stock pour ces deux types de zones. "stock(N)" représente le stock de beams blanks de nuance N.

Soit Zk une zone d'enfournement à chaud => Nuance NCC_k=Nuance NT_k= N

Alors : $stock(N) = stock(N) + (PCC_k - PT_k).(B_k - A_k)$

Soit Zk une zone d'enfournement à froid => Nuance NCC_k <> Nuance NT_k

Alors: 1) $stock(NCC_k) = stock(NCC_k) + PCC_k.(B_k - A_k)$

2) $stock(NT_k) = stock(NT_k) - PT_k.(B_k - A_k)$

L'état initial du stock est une donnée du problème. Après une zone, les règles ci-avant de calcul de variation de stock sur une zone nous renseignent un nouvel état du stock. Le parcours successif des zones nous donnent l'évolution du stock . A la fin de la production de la coulée continue et du train, nous aurons donc l'état final du stock.

Au premier chapitre, nous avons vu le calcul de l'enfournement à chaud pour une zone. En même temps que nous parcourons les zones pour le calcul du stock, il nous est possible de calculer l'enfournement à chaud. De cette manière, un seul parcours des zones définies par la coulée continue et du train nous est nécessaire.

D. Evolution de la structure des données par rapport au modèle 1.

Il faut tout d'abord souligner le fait que nous aimerions faire évoluer la structure du modèle 1 pour arriver au modèle 2, plutôt que d'inventer une nouvelle structure de données. Nous allons donc exprimer cette évolution par rapport aux structures de données du modèle 1.

a. Le train et la coulée continue.

Les structures de données du train et de la coulée continue ne sont pas modifiées. Le train est caractérisé par les tonnages auquels correspondent une productivité et une nuance (certaines nuances sont fictives : elles représentent les pauses). La coulée continue est caractérisée par un nombre de coulées et une nuance de chaque séquence de coulées.

b.Le stock.

Un nouvel élément entre en jeu : il s'agit du stock. Il lui faut donc une structure de données pour le représenter. Celle-ci est extrèmement simple : elle se compose d'une valeur de stock pour chaque nuance. Ce stock accompagne une instance de coulée continue. Il est le stock final résultant de l'évolution du stock depuis le stock initial (qui est une donnée) en fonction du déroulement du train et de la coulée continue. Il est inutile de retenir toute l'évolution du stock (ce qui peut être très coûteux point de vue espace mémoire) car d'une part, celle-ci peut être facilement retrouvée en partant du stock initial et en mettant le stock à jour en parcourant le train et la coulée continue en parallèle (nous avons vu dans la section précedente le moyen de retrouver cette évolution), et d'autre part, elle ne nous est pas utile.

c. Les modifications ou mouvements.

Nous reprendrons les mêmes mouvements que dans la version finale du modèle de notre algorithme du modèle 1. Nos modifications portent sur un tronçon de séquence(s) choisi aléatoirement dans les séquences de coulées de la coulée continue. Ce tronçon peut avoir une longueur d'une ou de deux séquences de coulées. Sur ce tronçon, on fait des changements de nuances et des changements du nombre de coulées. Ces premiers étaient faits en prenant les nuances qui se trouvent en face de la séquence à laquelle on attribuait justement cette nuance. On avait rectifié l'intervalle dans lequel on prenait les nuances de manière à ce qu'en cas de changement du nombre de coulées, certaines nuances ne soient pas oubliées. Les changements du nombre de coulées se faisaient en veillant à ne pas sortir de l'intervalle des min-max du nombre de coulées associées à une nuance.

Nous ajouterons quelques mouvements supplémentaires : la liste des nuances à essayer pour une séquence de coulées sera complétée d'une série de nuances. Celles-ci sont les nuances pour lesquelles le stock devient négatif à un moment donné de la coulée continue. Pourquoi ajouter ces nuances à celles qui se trouvent en face ? Pour cela, il nous faut analyser l'évolution du stock en fonction du train et de la coulée continue. Cette dernière est caractérisée par le fait que sa productivité est constante à tout moment tandis que la productivité du train varie très souvent. Plusieurs cas peuvent alors se présenter divisés en deux types :

1er type : cas où il y a de l'enfournement à chaud. (mêmes nuances)

1) Productivité du train > Productivité de la C.C. :

Le train demande, sur un même laps de temps, plus d'acier que la coulée continue ne peut lui en fournir : le reste est pris dans le stock.

$\Rightarrow$ le stock de cette nuance d'acier diminue.

2) Productivité du train = Productivité de la C.C.

Le train demande, sur un même laps de temps, exactement la quantité d'acier que la coulée continue peut lui en fournir.

$\Rightarrow$ le stock de cette nuance reste le même.

3) Productivité du train < Productivité de la C.C.

Le train demande moins d'acier, sur un même laps de temps, que la coulée continue ne peut lui en fournir.

$\Rightarrow$ le stock de cette nuance d'acier augmente.

2ème type : cas où il n'y a pas d'enfournement à chaud.

$\Rightarrow$ 2 nuances N_i (train) et N_j (coulée continue)

Le stock de N_i diminue de la quantité dont a besoin le train.

Le stock de N_j augmente de la quantité fournie par la coulée continue.

Ceci étant dit, il est évident que le stock peut augmenter tout aussi bien qu'il peut diminuer. On peut rencontrer des productions telles que le train réclame de l'acier que la coulée ne peut lui fournir, et, que le stock de cette nuance est vide.

Il faudrait alors modifier la coulée continue pour qu'elle remplisse le stock avant cette pénurie. En ajoutant ces nuances aux nuances en face, nous donnons la possibilité à notre algorithme de choisir ces nuances en vue de corriger une ou plusieurs pénuries. Il existera donc des endroits où le choix de la nuance ne se fera pas nécessairement pour maximiser l'enfournement à chaud mais pour renforcer un stock défectueux c'est-à-dire éviter une pénurie. Le critère de choix d'une modification n'est plus simplement la maximisation de l'enfournement à chaud mais aussi la réalimentation d'un stock pour qu'il n'y ait pas de pénurie.

Remarque :

On peut imaginer des séquences de montages au train telles que l'on ne puisse pas planifier la coulée continue pour satisfaire la condition de non-négativité du stock. En effet, par exemple, le premier montage du train est 400 tonnes de nuance 9 à du 200 tonnes par heure et le stock de cette nuance est vide. La coulée pourra produire de la nuance 9 à du 150 tonnes par heure maximum. Il manque donc 50x2=100 tonnes de nuances 9 durant les deux premières heures. Ce cas est considéré comme un cas d'infaisabilité pour notre modèle.

D. Le tabou appliqué : passage en revue des éléments du Tabou appliqué au problème du modèle 2.

Nous allons maintenant décrire les éléments du Tabou pour ce modèle 2. De nombreuses choses sont communes avec le modèle 1 : des références à cette même partie pour le modèle 1 seront faites en vue de montrer l'évolution des éléments du Tabou au travers des modèles.

a. Les solutions.

L'ensemble des solutions du modèle 2 n'est plus le même que celui du modèle 1. En effet, nous avons ajouté le stock qui fait partie d'une solution pour ce deuxième modèle.

S = ensemble des solutions au sens du modèle 1.

S_2 = ensemble des solutions au sens du modèle 2.

= (s , stock , pénalité)

où :

1) $s \in S$

2) "stock" est le stock final après la production de la coulée continue décrite dans "s" et la production au train.

3) "pénalité" est le nombre de fois que le stock d'une nuance devient négatif.

Remarque :

Le troisième élément du triplet qui définit une solution, à savoir "pénalité", est le nombre de fois que le stock d'une nuance devient négatif. Il est bon de préciser qu'à chaque fois que le stock d'une nuance devient négatif, le nombre de pénalité augmente de 1. Cela signifie qu'un stock d'une nuance particulière peut provoquer plusieurs pénalités. Par exemple, un stock devient négatif une première fois (penalité=1) puis redevient positif et redescend une nouvelle fois sous zéro : "penalité" vaudra deux (+ les pénalités pour les autres nuances).

Il est évident que ce nombre de pénalité est attaché à une planification de la coulée continue. Ce nombre de pénalités va servir à pénaliser la coulée continue pour que l'algorithme corrige ces problèmes de stocks négatifs. Les pénalités apparaîtront dans la fonction d'évaluation (voir plus loin).

b. Les règles de modifications des solutions ou mouvements.

Les règles de modification étant les mêmes que pour la version finale du modèle 1, nous ne répéterons que la définition formelle du voisinage. Il vous est possible de consulter la section G du chapitre 2 pour avoir plus de renseignements.

Pour les tronçons de deux séquences de coulées, le voisinage généré est :

$N(s) = \{ s' \mid \exists k \in [1,n-2]$ tq :

$$\forall i \neq k, k+1 : (c_i,N_i) = (c'_i,N'_i) \quad (1)$$

$$N'_k, N'_{k+1} \in \text{Nuances\_en\_face} \quad (2)$$

$$c'_k \in [\text{borne\_inf}_{N_k}, \text{borne\_sup}_{N_k}] \quad (3)$$

$$c'_{k+1} \in [\text{borne\_inf}_{N_{k+1}}, \text{borne\_sup}_{N_{k+1}}] \quad (4) \ \}$$

Pour les tronçons d'une séquence de coulées, le voisinage généré est :

$N(s) = \{ s' \mid \exists k \in [1,n-1]$ tq :

$$\forall i \neq k : (c_i,N_i) = (c'_i,N'_i) \quad (1)$$

$$N'_k \in \text{Nuances\_en\_face} \quad (2)$$

$$c'_k \in [\text{borne\_inf}_{N_k}, \text{borne\_sup}_{N_k}] \quad (3) \ \}$$

c. La fonction d'évaluation.

La fonction d'évaluation est différente par rapport au premier modèle. En effet, il faut tenir compte ici du stock et de la possibilité de pénurie. Cette dernière n'étant pas acceptable pour une solution faisable, nous nous devons d'indiquer à l'algorithme qu'il est sur une solution inacceptable.

Nous avons parlé des stocks négatifs et du troisième élément du triplet. Il nous faut donc repérer les stocks qui deviennent négatifs à un moment donné. Lors du calcul du stock et de l'enfournement à chaud, nous observerons les stocks que nous avons modifiés après chaque zone, de telle façon que si l'un deux devient négatif, l'élément "pénalité" de la solution soit incrémenté de un. Nous aurons

donc le nombre total de stock négatif à la fin du parcours des zones que définissent la coulée continue et le train.

A la valeur de l'enfournement à chaud, nous retirons le poids des pénalités de telle façon que la solution qui comporte des stocks négatifs donne un résultat moins bon. L'expression de la fonction objective est la suivante :

fonction objective = enfournement à chaud - #pénalité . valeur_pénalité

où valeur_pénalité est un nombre entier assez grand que pour la pénalisation fasse son effet.

Cette valeur de pénalité peut être plus ou moins définie théoriquement mais une expérimentation s'impose. Théoriquement, on aurait tendance à mettre une pénalité qui donne un résultat négatif à la fonction objective pour être sûr que des solutions qui donnent un faible enfournement à chaud mais pas de stock négatif, soit prise plutôt que celles avec un enfournement à chaud élevé et des stocks négatifs.

d. Les conditions taboues.

En ce qui concerne les conditions taboues, nous reprenont celles qui ont été définies dans la version finale du modèle 1 (paragraphe H - chapitre I). Pour rappel, il y a deux conditions taboues pour une modification portant sur un tronçon de deux séquences de coulées et une condition taboue sur un tronçon d'une séquence de coulées.

e. Fonction d'aspiration.

La fonction d'aspiration est définie par un seuil. Celui-ci est la valeur de la fonction d'évaluation prise en la meilleure solution du moment.

f. Nombre de détériorations successives avant de s'arrêter.

Comme le comportement de notre modèle nous est inconnu avant de l'expérimenter, il nous est impossible de fixer cette valeur définitivement. Pour débuter, nous fixerons celle-ci à 200. Nous ferons des essais avec des valeurs de kmax plus grandes par la suite.

E. Chronologie du développement.

Toujours dans un but de mettre en évidence la démarche de développement d'une méthode et de montrer l'intérêt de développer plusieurs stratégies, nous allons expliciter tout le développement de notre deuxième modèle.

Nous avons commencé notre développement avec un algorithme presque semblable à la version finale du modèle 1 (Version 6). Nous y avons ajouté le calcul du stock, et, modifié le voisinage généré et la fonction objective.

Nous avons placé le calcul du stock dans la procédure de calcul de la fonction objective; ceci a été expliqué dans le paragraphe consacré au calcul du stock.

Dans le modèle 1, nous générions notre voisinage en modifiant un tronçon choisi aléatoirement dans la séquence des coulées de la coulée continue. Ce tronçon comportait une ou deux séquences de coulées. D'une part, ces modifications portaient sur le nombre de coulées de la ou les séquences de coulées dans les limites des bornes min-max données par les nuances. D'autre part, nous modifions la ou les nuances avec les nuances des montages du train se trouvant en face du tronçon. Dans ce modèle 2, nous ajoutons à ces nuances celles, parmi toutes les nuances existantes, dont le stock tombe au moins une fois en pénurie durant le déroulement du train (et de la coulée continue).

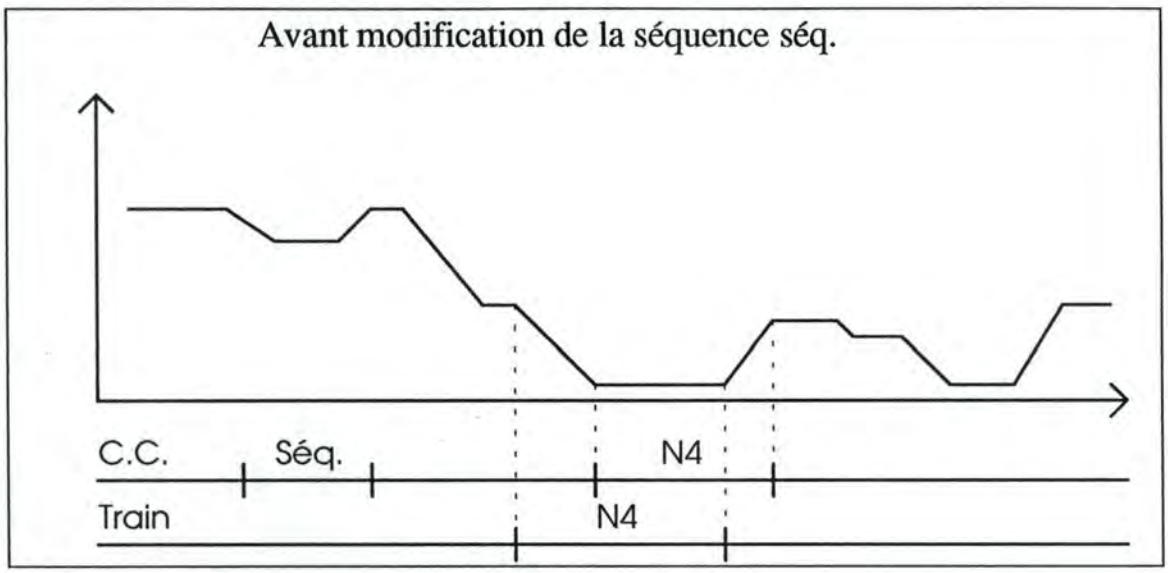
La fonction objective tient maintenant compte des stocks négatifs en plus de la valeur de l'enfournement à chaud. On pénalise l'enfournement à chaud en diminuant sa valeur par le nombre de stock(s) négatif(s) trouvé(s) multiplié par une valeur de pénalité (nous en parlerons plus loin).

Nous avons essayé de tenir compte des tonnages en négatif plutôt que de tenir compte du nombre d'endroits de pénurie. En pénalisant les stocks négatifs, notre modèle a plutôt tendance à corriger les stocks des nuances pour lesquelles le nombre de pénuries sur la longueur de la coulée continue est le plus grand. Tandis qu'en pénalisant les tonnages, il aura plutôt tendance à corriger les stocks des nuances pour lesquelles le tonnage en négatif (l'importance de la pénurie) est le plus grand. Ce sont deux philosophies bien différentes !

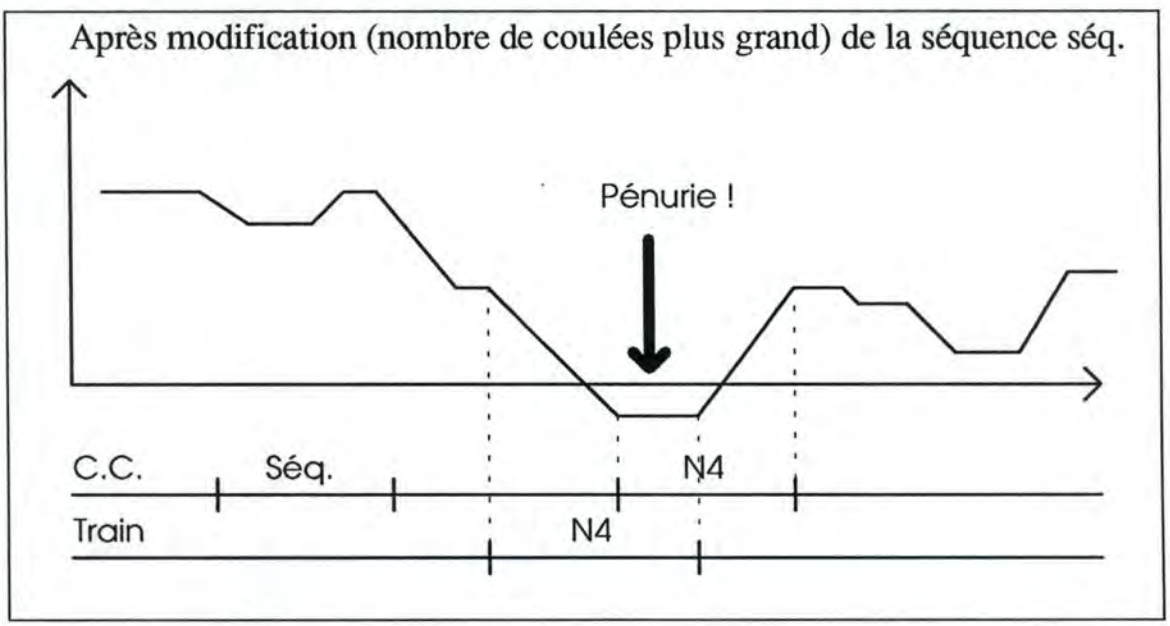
Quelle que soit la manière dont on pénalise les pénuries, cette version a le désagréable inconvénient de décaler une partie de la coulée continue lorsque le ou les nombres de coulées de la ou les séquences de coulées sont modifiées. Ce type de modification peut entraîner non seulement des pertes d'enfournement à chaud mais en plus des bouleversements dans les stocks.

C'est pourquoi nous sommes passés à un modèle avec des tronçons composés tous de deux séquences de coulées et dont les modifications doivent laisser invariante la somme des deux nombres de coulées. De cette façon, nous évitons

des bouleversements dans le stock comme l'illustrent les dessins suivants. Dans le premier dessin, nous avons représenté la courbe de stock de la nuance N4 en fonction de la coulée continue et du train. Nous voyons qu'il n'y a aucun problème de pénurie.



Une modification du nombre de coulées de la séquence "séq" provoque le décalage de la partie droite de cette séquence comme l'illustre le dessin suivant.



A partir de ces deux dessins, nous voyons clairement qu'une modification d'une séquence peut faire apparaître des bouleversements dans le stock, même si cette modification se passe bien avant l'endroit où la pénurie se produit

Dans notre modèle 1, ces bouleversements se produisent uniquement sur la valeur de l'enfournement à chaud. Tandis que dans le modèle 2, nous venons de voir que ceux-ci pouvaient se produire non seulement sur la valeur de

l'enfournement à chaud mais aussi sur les stocks. Le but de notre modèle 2 est bien sûr de maximiser l'enfournement à chaud mais aussi d'éviter les pénuries. Ces deux objectifs vont parfois l'un à l'encontre de l'autre.

Ensuite, nous avons adapté la valeur de la pénalité. Nous avions dit qu'une pénalité plus grande était sûrement préférable pour éviter de prendre des solutions qui donnent un résultat apparemment bon mais en fait, résultant d'un grand enfournement à chaud mais de stocks négatifs trop peu pénalisés. L'expérimentation nous a montré qu'une valeur de pénalité faible est nettement préférable. Ce paradoxe sera expliqué dans l'analyse des résultats.

Malgré tout, il ne faut tout de même pas prendre une pénalité trop faible au risque de perdre l'influence de celle-ci dans la résolution des pénuries des stocks. En effet, une pénalité faible courra le risque de voir l'algorithme préférer de l'enfournement à chaud plutôt que la résolution d'une pénurie. Je suggère de choisir une valeur de pénalité supérieure ou égale à 500. Cette valeur peut paraître insuffisante mais nous allons expliquer le pourquoi de celle-ci. Pour cela, il nous faut être un peu technique.

Par simplicité d'expression, nous augmentons de un le nombre de stocks négatifs à chaque fois qu'un stock calculé est négatif. Autrement dit, après chaque zone (voir paragraphe de calcul du stock), on calcule le stock de la ou les nuances suivant que l'on a de l'enfournement à chaud ou non. Si un stock est négatif, nous incrémentons de 1 le nombre de stocks négatifs. Ce qui signifie qu'une pénurie peut être comptabilisée plusieurs fois. Il est même à peu près certain qu'elle le sera au moins deux fois. Donc, la pénalité pour une pénurie ne sera pas pénalisée de 500 mais d'au moins 2 fois 500. Cette valeur est supérieure à ce que l'on gagne en enfournement à chaud sur une séquence.

Nous avons ensuite transformé notre algorithme pour l'adapter à des nombres de nuances plus grands c'est-à-dire pour un nombre de nuances allant jusque 45.

Quelques tests supplémentaires ont été effectués avec des tronçons de longueurs 3. L'idée vient du fait qu'il faut peut-être que le tronçon soit proportionnel au nombre de séquences de la coulée continue.

Remarque :

Nous n'avons pas précisé le mode de calcul de l'enfournement à chaud. Nous calculerons celui-ci sur toute la coulée continue même dans la version dont la longueur du tronçon reste constante. En effet, nous y sommes contraint à cause du stock. Nous devons calculer l'évolution du stock pour chaque nuance sur toute la coulée continue car les effets des modifications même sur un tronçon de longueur constante peuvent se produire ailleurs que sur ce tronçon. Reprenons

les dessins de la page précédente. Nous remarquons que le décalage d'une séquence de N4 provoque une pénurie. Si nous remplacions cette nuance N4 par N9, la courbe continuerait à descendre jusqu'à la fin du montage sur le train. La partie de droite s'en trouverait décalée vers le bas, ce qui d'après le dessin provoquerait une deuxième pénurie ou une utilisation des stocks au lieu d'un enfournement à chaud : le recalcul est donc nécessaire.

F. Problème de testing du modèle 2.

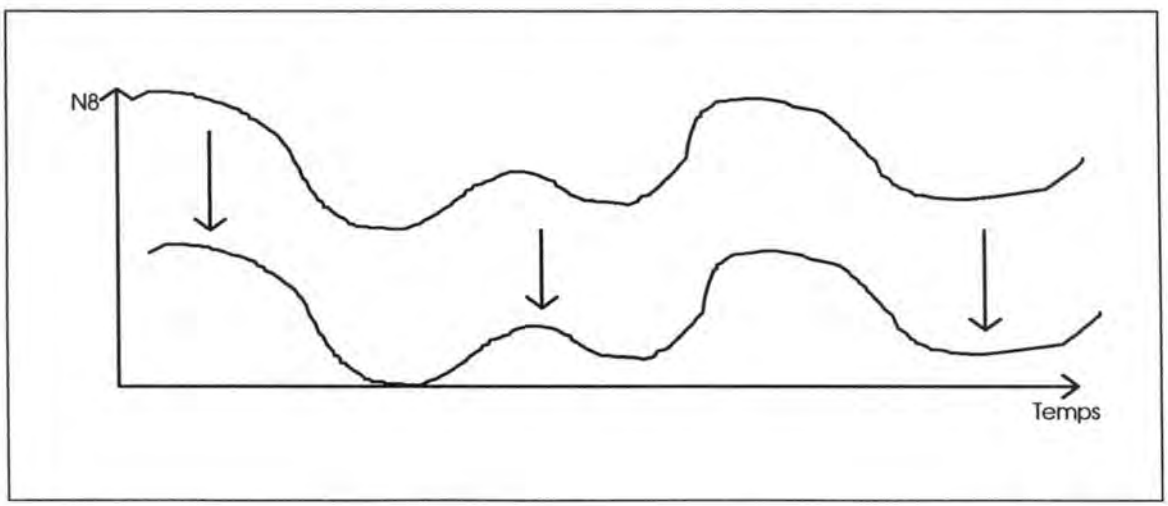
Dans le modèle 1, il nous suffisait de construire des productions pour le train et de "construire" avec notre algorithme une planification de la coulée continue qui maximisait notre enfournement à chaud. Le modèle 2 comporte un élément supplémentaire : le stock. Il faut donc lui donner un état initial pour tester le comportement de notre algorithme.

Pour le modèle 1, nous nous étions construit une heuristique qui nous donnait la production de la coulée continue fournissant un enfournement à chaud maximum. Pour ce modèle 2, il nous est impossible de construire une heuristique qui nous permette de calculer l'enfournement maximum pour un état initial des stocks et une coulée continue donnée. De ce fait, il nous était impossible de tester le modèle 2 de la même manière que le modèle 1.

Nous avons alors construit une méthode pour tester notre algorithme, pour laquelle nous étions sûr d'avoir un résultat que nous connaissions. Nous allons maintenant expliquer cette méthode.

Dans un premier temps, nous avons fait exécuter notre algorithme avec un stock suffisamment grand pour être sûr qu'il n'y aurait jamais de pénurie. Pour être sûr de cela, nous avons observé les besoins du train pour chaque nuance. Nous en avons conclu qu'il fallait prendre un stock initial de 10000 pour toutes les nuances. L'exécution nous a donné les courbes des stocks desquelles nous pouvions, pour chaque nuance, examiner le minimum.

Ensuite, nous avons exécuté une deuxième fois notre algorithme avec un stock construit de la manière suivante. La valeur du stock pour une nuance est la différence entre 10000 et la valeur minimum de la courbe de son stock. Cela équivaut à "descendre" la courbe de stock de la nuance de telle manière à ce que son minimum touche l'axe horizontal mais aussi que la courbe ne passe pas en



dessous de 0. De cette façon, la solution que nous avions trouvée en exécutant notre algorithme avec un stock à 10000 est toujours valable. Le dessin suivant nous montre, pour une nuance, la "descente" de la courbe.

Cela peut paraître stupide car on pourrait croire qu'il va nous donner systématiquement la même solution ! Pour bien comprendre que cela n'est pas facile, nous allons examiner notre algorithme. Celui-ci part d'une solution initiale quelconque : la planification initiale de la coulée continue est choisie totalement au hasard. Cela veut dire qu'on ne se préoccupe pas du tout de construire une solution initiale comportant un minimum de pénuries. Donc, à partir de cette solution initiale comportant éventuellement des pénuries, notre algorithme devra construire une solution sans pénurie en les éliminant progressivement par le jeu des pénalités et en construisant un certain enfournement à chaud. En conclusion, il va devoir reconstruire, en partant d'une solution comportant des pénuries, une solution optimale que nous connaissons. Nous avons donc une manière tout à fait valable de tester notre algorithme de notre modèle 2 !

G. Remarque importante.

Nous avons déjà dit qu'à chaque nuance sont associées des bornes min-max du nombre de coulées. C'est-à-dire que lorsque nous choisissons une certaine nuance pour une séquence, nous sommes obligés de choisir un nombre de coulées compris entre ces bornes. Dans le modèle 1, nous avions pris ces bornes égales à 8 et 11 pour toutes les nuances.

Nous avions reçu ces valeurs lorsque nous avons commencé à considérer ce problème. Nous avons été invité à changer ces bornes suite à des modifications dans le problème industriel. On nous avait bien spécifié que toutes les données de départ étaient susceptibles d'évoluer au fur et à mesure de l'étude du problème. C'est pourquoi les bornes ne sont plus 8 et 11 pour toutes les nuances mais 3 et 5. Il est évident que le problème n'est plus le même !!

En réalité, nous avons reçu 3 et $+\infty$ comme valeurs pour ces bornes. Mais l'infini n'est pas formalisable sur un ordinateur. Il a donc fallu trouver une astuce de représentation. Nous avons mis 3 et 5 à la place et nous allons montrer pourquoi cela revient au même. Nous pouvons donc avoir 3,4,5,6,7,8,.... comme nombre de coulées pour une séquence de coulées. Nous représentons 3,4 et 5 comme une séquence de coulées de 3,4 et 5. Pour représenter le 6, nous employons 2 séquences de coulées auxquelles nous attribuons 3 comme nombre de coulées; c'est bien un nombre de coulées acceptables. Un nombre de coulées de 7 se représente sur 2 séquences de coulées en attribuant 4 et 3 coulées comme nombre de coulées de celles-ci. Et ainsi de suite pour les autres nombres où nous devrons passer à plus de 2 séquences pour représenter un nombre de coulées supérieur à 10.

Le lecteur devrait s'interroger quelque peu sur cette valeur infinie du nombre maximum du nombre de coulées. En effet, il a été dit que cette borne maximum était la limite supérieure d'utilisation d'un répartiteur. Cela voudrait-il dire qu'un répartiteur peut être utilisé indéfiniment ? La réponse est non bien entendu. Il y a eu un changement dans le procédé de remplacement d'un répartiteur. Le processus est le suivant : sur un axe pouvant tourner sur lui-même sont fixés deux répartiteurs; lorsqu'un répartiteur est "usé", on change de répartiteur en faisant tourner l'axe et le second répartiteur continue le travail. Tout se passe donc comme si on continuait dans un même répartiteur.

Nous allons maintenant voir ce que ce changement de borne (3-5) a comme influence sur notre problème. Comme "5 coulées" est un nombre inférieur à "8 coulées", il nous faudra beaucoup plus de séquences de coulées pour produire à la coulée continue durant une période équivalente à la période de laminage du train. Pour une production de 5 jours au train (5 x 24 heures = 120 heures), il nous faut maintenant 40 séquences de coulées de 3 coulées (40 x 3 heures = 120

heures) à la place de 15 séquences de coulées de 8 coulées (15 x 8 heures = 120 heures).

L'espace des solutions s'en trouve agrandi en conséquence. En effet, en ne considérant que les nombres de coulées, nous avons maintenant 3^{40} solutions à la place de 4^{15} solutions. Quelques calculs pour nous éclairer ! Supposons que nous ayons 15 nuances dans notre problème. Le tableau suivant indique le nombre de solutions de l'ensemble X dans le cas 3-5 et dans le cas 8-11, pour 5 et 10 jours.

	5 jours	**10 jours**
3-5	$1{,}3 \cdot 10^{66}$	$1{,}8 \cdot 10^{132}$
8-11	$4{,}7 \cdot 10^{26}$	$2{,}2 \cdot 10^{53}$

Nous voyons que le cas 3-5 en 5 jours a un nombre de solutions presque équivalent au cas 8-11 en 10 jours. Il apparaît clairement en observant les résultats du modèle 1, que l'erreur est fonction du cardinal de X. Observons alors l'erreur moyenne pour ce dernier cas. L'erreur se situait au environ de 2,9 %. Nous devrions obtenir le même taux d'erreur avec le cas 3-5 en 5 jours vu que nous avons plus ou moins le même nombre de solutions. Tout d'abord, nous avons pu faire tourner notre heuristique pour trouver les optimums des tests car il est évident qu'ils ont changés. Les résultats sont dans le tableau suivant.

Test	**Optimum**
1	10984
2	11611
3	13835
4	12925
5	9694
6	9646
7	4879
8	5035
9	21363
10	22521
11	27441
12	25942
13	19310
14	19489
15	10071
16	10104

Nous pouvons déjà nous apercevoir que les valeurs ont nettement augmenté par rapport au cas 8-11. Nous ne les avons pas remises pour ne pas embrouiller les esprits. Mais nous pouvons expliquer cette augmentation de l'enfournement à chaud. Les séquences de coulées en 3-5 sont beaucoup plus petites qu'en 8-11 et donc s'affinent plus facilement avec le train. Dans le cas 8-11, une grande partie de la séquence provoquait de l'enfournement à froid car il n'existe pas une partie de montage d'une même nuance au train qui soit supérieur à 3 heures. Tandis que nous obtenons un enfournement à chaud sur une plus grande partie de la séquence dans le cas 3-5.

Nous avons fait tourner pour les six premiers tests notre dernière version (version 6) de notre algorithme en cas 3-5 pour analyser les erreurs. Le tableau suivant nous montre les résultats en erreur par rapport à l'optimum.

Test	Optimum	Version 6 - Modèle 1	Erreur
1	10984	10210	7 %
2	11611	10432	10 %
3	13835	12925	6 %
4	12925	12202	5 %
5	9694	9429	2 %
6	9646	9093	2 %

Nous avons une erreur moyenne de 5,3 %. Nous voyons donc qu'une erreur plus grande survient avec le cas 3-5. Nous avons tester les versions 4 et 5 de notre modèle 1 en cas 3-5. La version 4 est extrêmement médiocre avec une erreur moyenne de plus de 10 %. La version 5 est meilleure avec 4 % mais toujours loin des résultats que l'on obtenait avec la version 6 dans le cas 8-11.

Nous avons alors modifié notre voisinage. Nous faisons des modifications sur un tronçon de 3 séquences de coulées. L'idée est que comme le nombre de séquences est plus grand , le tronçon doit être adapté. Les résultats sont alors bien meilleurs comme le montre le tableau suivant. Cette nouvelle version de modèle 1 est la version 7.

Test	Optimum	Version 6 - Modèle 1	Erreur
1	10984	10763	2 %
2	11611	11112	4,2 %
3	13835	13454	2,7 %
4	12925	10901	0,1 %
5	9694	9663	0,3 %
6	9646	9612	0,3 %

La moyenne est maintenant de 1,6 % d'erreur par rapport à l'optimum. On peut donc conclure en disant que le tronçon doit être plus ou moins proportionnel au nombre de séquences de coulées de la coulée continue.

Remarque :

Bien que la version 7 soit la meilleure , nous commencerons le développement de notre modèle 2 avec comme base la version 6 de notre modèle 1, version qui s'est avérée la meilleure à ce moment. Nous essayerons bien entendu d'adapter la version 7 à notre modèle 2.

Nous pratiquons de la sorte pour suivre l'ordre chronologique dans lequel se sont passés les événements. En effet, nous ne nous étions pas trop préoccupés de connaître les résultats de notre modèle 1 pour le cas 3-5 avant d'entamer le développement du modèle 2. Ce n'est que par après que nous avons fait ces constatations.

H. Résultats et analyse de ce développement.

Nous allons examiner les résultats de notre développement en leurs donnant une explication. Les résultats proviennent de l'exécution de notre modèle 2 sur les 16 tests qui ont été décrits dans le modèle 1 (voir section F du chapitre I). Les bornes min et max du nombre de coulées d'une séquence sont toutes de 8 et 11 quelle que soit la nuance. Nous avons expliqué notre principe de test pour notre modèle 2. C'est pourquoi nous retrouverons dans chaque tableau de résultats au moins les deux colonnes suivantes : une indiquant les résultats avec un stock infini (stocks égale à 10000 pour toutes les nuances) et une indiquant les résultats avec stock limité (voir la section F du chapître III indiquant comment ces valeurs ont été obtenues pour chaque nuance).

a. Résultats numériques.

Version 1.

Cette première version est la même que la version finale de notre modèle 1 excepté la génération du voisinage, le calcul du stock en plus dans la fonction objective (y compris le calcul du nombre de nuances négatives) et l'ajout d'une partie pénalité dans la fonction objective.

Cette version s'est avérée un échec total. En effet, presqu'aucune réponse acceptable n'est retrouvée (il nous laisse des pénuries) avec cette technique de génération de voisinage, quelle que soit la valeur de la pénalité. Les quelques solutions acceptables trouvées ne nous redonnaient jamais la réponse fournie avec le stock infini et de plus, la réponse donnée était extrêmement mauvaise du point de vue enfournement à chaud.

Nous ne donnerons pas de tableau de résultats puisque celui-ci devrait se composer de quelques cases avec des résultats et essentiellement des barres signifiant qu'on n'a pas trouvé de solution acceptable.

Version 2.

Dans cette version, nous allons tenir compte des tonnages des stocks négatifs et plus du nombre de pénuries. Le reste de l'algorithme reste le même.

Encore une fois, cette version nous conduit à un échec total. Les résultats sont comparables à ceux donnés par la version 1.

Il apparaît de ces deux versions que le problème ne se situe pas dans la définition de la fonction objective mais plutôt dans la définition du voisinage. Nous allons donc essayer d'y remédier.

Version 3.

Cette version 3 va s'occuper de trouver la solution au problème de génération du voisinage. Nous allons prendre des tronçons de longueur deux uniquement. Cette longueur (somme des deux nombres de coulées) reste constante au cours d'une modification. Nous avons commencé les expérimentations avec une valeur de pénalité de 5000.

Le tableau suivant nous montre les résultats de cette version. Les barres signifient que l'on a pas trouvé de résultat.

Test	Résultats avec stock infini	Résultat avec stock limité
1	9740	8565
2	9868	8074
3	12609	11732
4	11595	9955
5	8160	8320
6	8331	7711
7	4243	4243
8	4048	3487
9	19080	18893
10	20103	-
11	25542	25351
12	24364	23343
13	17100	-
14	17062	15384
15	8765	-
16	8422	-

Nous pouvons observer que 4 tests ne nous donnent pas de solution acceptables au bout des 5 essais. L'algorithme ne résout donc pas toutes les pénuries. Ensuite, nous voyons pour les autres tests que les résultats avec stock limité sont pratiquement toujours différents de ceux avec stock illimité excepté le test 7 qui nous donne la même valeur et le test 5 qui nous donne une valeur supérieure.

Ces résultats ne nous satisfont pas vraiment car non seulement ceux-ci diffèrent fortement par rapport à ceux qu'on devraient trouver mais en plus il existe des tests qui n'ont pas de solution acceptable.

Nous pouvons expliquer l'échec dans certains tests par la grandeur de la pénalité. En effet, cette valeur de pénalité est peut-être si grande que des sauts entre solutions acceptables ne peuvent s'effectuer. C'est pourquoi nous allons ajuster cette valeur de pénalité.

Version 4.

La seule différence entre cette version et la précédente est la valeur de la pénalité qui est nettement inférieure. Nous espérons ainsi une plus grande souplesse de la part de notre algorithme et ainsi nous assurer qu'une solution acceptable puisse au moins être trouvée. Nous espérons aussi que les résultats sont moins éloignés de ceux attendus. Le tableau suivant nous montre les résultats des 5 essais ainsi que le résultat trouvé avec un stock infini. Les 5 essais sont accompagnés (colonne juste à leurs droites) du nombre de pénuries restant à la fin de l'exécution de l'algorithme. La valeur de la pénalité a été fixée à 500.

Test	Avec stock ∞	Essai 1		Essai 2		Essai 3		Essai 4		Essai 5	
1	9740	**9740**	0	**9441**	0	**9740**	0	8453	1	8851	1
2	9868	9315	1	8507	4	**9868**	0	9618	1	**9366**	0
3	12609	**12009**	0	**12609**	0	**12609**	0	**12609**	0	**12609**	0
4	11595	**11595**	0	**11595**	0	11108	1	11255	4	11346	1
5	8160	7467	1	**8160**	0	6518	4	**8160**	0	7540	2
6	8331	**7675**	0	**8331**	0	7856	1	**8319**	0	**7920**	0
7	4243	**3924**	0	**4243**	0	**4215**	0	**4072**	0	**4227**	0
8	4048	**2893**	0	**3487**	0	3665	1	3153	1	3154	2
9	19080	**19157**	0	17301	1	**18868**	0	**18481**	0	**18104**	0
10	20103	17535	7	**18247**	0	19069	2	18492	6	**18456**	0
11	25542	23405	3	24515	2	23925	3	**25351**	0	24337	2
12	24364	23658	4	24016	2	**22503**	0	20724	6	18108	9
13	17100	**17123**	0	**16621**	0	13921	2	16124	1	15574	3
14	17062	**15873**	0	14736	4	15003	2	**15878**	0	12829	4
15	8765	**7824**	0	**8516**	0	**8036**	0	**7793**	0	**8424**	0
16	8422	4362	3	7121	1	7286	1	7003	1	**7320**	0

Nous observons de suite que les résultats sont nettement meilleurs maintenant avec une pénalité plus faible. En effet, dans les 7 premiers tests, nous obtenons un résultat (cases ombrées) qui est le même que celui obtenu avec un stock infini. De plus, certains tests nous le donnent plusieurs fois au cours des essais.

De plus, les autres tests nous donnent au moins une fois un résultat sans pénurie (cases avec résultat en gras) et le meilleur résultat de ces tests s'éloigne de moins de 5 % de la valeur avec un stock infini. Nous pouvons nous estimer heureux de ces résultats et conclure que notre algorithme sait résoudre les pénuries par le jeu des pénalités à condition de bien choisir la valeur de cette pénalité.

Nous allons expliquer la différence de comportement entre ces deux versions : celle avec une valeur de pénalité élevée et celle avec une valeur de pénalité faible. Pour cela, nous allons nous inventer une petite histoire bien concrète qui va nous illustrer le comportement de notre algorithme.

Imaginons des montagnes dont certains sommets dépassent la couche de nuages. Il y a des gens qui se promènent parmi celles-ci. Ils voudraient voir le soleil. Certains sont très pressés et d'autres le sont moins. Ces premiers montent sur le premier pic qu'ils rencontrent sans se soucier de voir s'il dépasse les nuages ou si il est très haut. Les seconds, très calmes et patients, se promènent au milieu des montagnes pour trouver celle qui leur paraît la plus haute et qui dépasse bien les nuages. Le résultat est le suivant : les premiers seront déçus car soit ils auront trouvés un pic qui dépasse les nuages mais faiblement, soit un pic qui ne dépasse pas les nuages. Les seconds, eux, auront sûrement trouvés un pic qui dépasse les nuages mais en plus ils seront très probablement sur le plus haut pic.

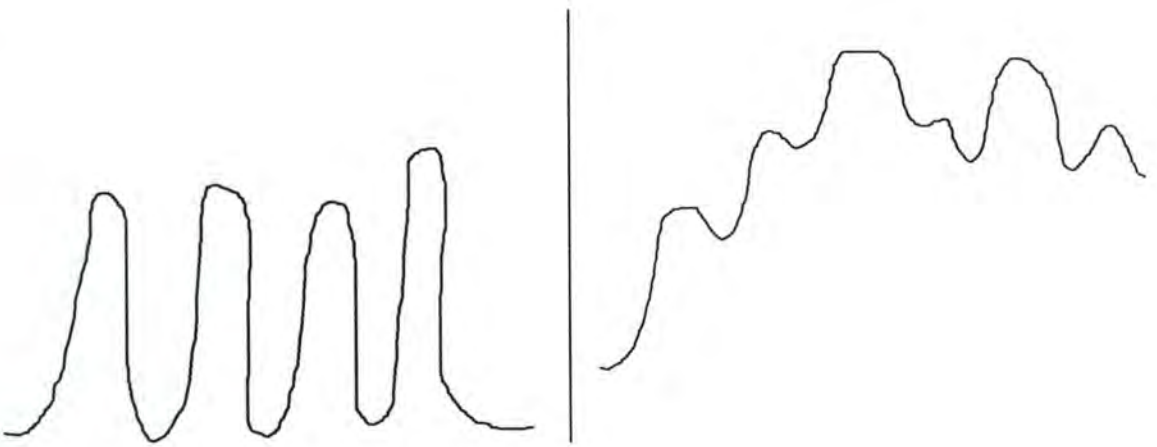
Nous pouvons comparer les montagnes avec notre fonction objective, les deux types de gens avec les deux types d'algorithme (à grande et à faible valeur de pénalité). Ainsi, les algorithmes avec une forte valeur de pénalité (Algorithme A1) correspondent aux gens pressés et ceux avec une faible valeur de pénalité (Algorithme A2) correspondent aux gens calmes et patients.

L'algorithme A1 essaye de résoudre les pénuries le plus vite possible; il monte directement sur une des premières montagnes qu'il trouve sans se soucier de sa hauteur. Donc, il ne se soucie pas de construire un enfournement à chaud. Il a parfois la chance d'avoir pris une montagne de hauteur moyenne et donc de sortir des nuages : il trouve parfois une solution acceptable. Nous obtenions quelques solutions acceptables dans la version 3.

L'algorithme A2 résout aussi les pénalités mais bien à son aise. Il trouvera donc une grande montagne (la plus grande parfois) sans se précipiter sur la première venue. Il aura donc construit un enfournement à chaud convenable et résolu les pénuries. Ce sont les résultats de la version 4.

Le piège pour ces algorithmes est de prendre une jeune montagne toute effilée et pas encore très haute mais à partir de laquelle on ne sait pas sauter vers une autre car le vide entre les deux pics est très important. Il vaut mieux prendre une vieille montagne bien haute et large à partir de laquelle on peut passer de l'un sommet à un autre voisin. Les algorithmes à grande valeurs de pénalité auront la

forme des jeunes montagnes (une pénalité en plus ou en moins fait varier fortement la fonction objective) et les algorithmes à faibles valeur de pénalité auront la forme des vieilles montagnes (faible variation de la fonction objective lorsque le nombre de pénalité varie de un)..



Il reste le cas d'une pénalité très faible. Il correspond aux gens qui ne sauraient jamais se décider, peur de se tromper de plus haut sommet. Ils resteront toujours sous les nuages et donc, ces algorithmes ne trouveront jamais de solution acceptable.

Version 5.

Nous avons essayé de prendre des tronçons de 3 séquences de coulées. Les résultats avec stocks infinis, nous l'avons vu dans le paragraphe G, sont bien meilleurs. Par contre, lorsque nous résolvons des pénuries, il y a un terrible problème de temps d'exécution. En effet, il faut se rappeler comment on génère nos modifications. Pour une modification de la première séquence de coulées du tronçon et pour une modification de la deuxième séquence de coulées du tronçon, nous faisons toutes les modifications de la troisième séquence de coulées du tronçon. Il y a donc une explosion de la taille de notre voisinage lorsque nous passons d'un tronçon de deux séquences de coulées à un tronçon de trois séquences de coulées. De plus, nous ajoutons les nuances pour lesquelles le stock devient négatif (au moins une fois) au cours de la production, à toutes les nuances à essayer et cela pour chaque liste de nuances associées à une séquence de coulées. Cela fait encore exploser notre taille de voisinage. Le temps consacré à une itération est très long et de ce fait, le temps total d'exécution est très long.

Nous n'avons donc pas fait de tests pour cette version par manque de temps évident !

Version 6 :

Cette dernière version n'est pas là pour améliorer quoi que ce soit mais pour observer notre algorithme lorsque nous considérons le problème avec un stock de 45 nuances différentes. Il n'y a bien sûr pas 45 nuances par montage de profil mais nous pouvons utiliser au cours des 10 jours (nombre de jours qui nous était aussi demandé) 45 nuances différentes.

D'où vient ce nombre de 45 nuances ? Nous avons expliqué qu'il y avait plusieurs taille de profils (presque une infinité même entre 0 et ± 1 mètre). Pour laminer ces profils, il nous faut 3 types de beams blanks : des BB0, des BB1 et des BB2. Ce qui les distinguent est leur section. En fonction du profil, on choisira un beam blanks particulier. Cependant, au cours d'une même semaine, nous laminerons des profils issus d'un même type de BB. Malgré tout, il nous faut des réserves de tous les types de BB. Comme il y a 15 nuances différentes et 3 types de BB, nous obtenons 45 "nuances" (le nom ne convient plus) de BB qui se distinguent soit par la qualité de leur acier, soit par la section du BB.

Nous avons construit un seul test (test 17) de caractéristiques décrites ci-dessus. Nous avons fait 5 essais dont voici les résultats accompagnés du nombre de pénalité(s) restante(s) après la fin de l'exécution du programme. Le résultat avec stock infini est de 9832.Ces 5 essais nous conduisent à une solution acceptable.

Avec stock limité	
6698	4
6356	5
7260	3
7780	0
6899	2

Remarque : [*temps d'exécution et nombre d'itérations*]

Nous n'avons pas parlé des temps d'exécution et du nombre d'itérations car nous étions trop préoccupé par le fait de trouver un algorithme qui nous corrigeait nos pénuries. En fait, nous en avons seulement parlé dans la version 5 où le temps devenait un problème.

Nous pouvons dire que 3 à 5 minutes sont nécessaires pour exécuter notre version 4. En général, moins de 10 minutes suffisent pour tous les tests. Cependant, il faut nuancer ces résultats car les temps sont extrêmement instables d'un essai à un autre et d'un test à un autre. Nous avons pu observer que plus de 1000 itérations au total étaient parfois nécessaires pour certains essais de

certain test alors que moins de 400 itérations suffisait pour certain essai de certain test. C'est pour cela que nous avons dit qu'il nous fallait en général moins de 10 minutes pour l'exécution de notre algorithme avec un "kmax" de 200.

Ce nombre "kmax" a été porté à 1000 pour certaines versions de notre modèle 2 : c'est le cas des premières versions. Le but était de voir s'il ne le fallait pas plus de temps pour se stabiliser. Nous avons vu que ce n'était pas le cas.

I. Etude des conditions taboues et de la fonction d'aspiration.

Nous allons étudier les variations de résultats (au sens large) lorsque nous modifions les conditions taboues dans notre modèle 2. Nous y apporterons aussi une explication.

Nous utiliserons les mêmes types de conditions taboues que dans l'étude similaire qui a été réalisées pour le modèle 1. Les deux tableaux suivants sont les résultats pour le même test (test 1),le même générateur de nombre aléatoire mais **dans le cas 3-5.** Les stocks sont les mêmes que ceux construits pour tester le modèle 2.

Tableau 1 :

Lg liste Tabou	Résultats	Nbre iter.	Temps	# sol. taboues	# aspiration
2	9740	519	50/33	0	0
14	9740	519	48/32	0	0
40	9740	519	47/33	0	0
100	9740	519	49/34	1	0
1000	9740	519	60/42	36	0
5000	9740	519	119/87	36	0

Tableau 2 :

Lg liste Tabou	Résultats	Nbre iter.	Temps	# sol. taboues	# aspiration
2	9740	519	48/32	0	0
14	9740	519	49/33	15	0
40	9740	519	48/33	20	0
100	9740	519	51/34	41	0
1000	9740	519	63/43	132	0
5000	9740	519	122/89	132	0

Nous remarquons que nous obtenons 9740 quelle que soit la longueur de la liste Tabou mais en plus le nombre d'itérations reste maintenant constant et les nombres de solutions taboues et d'aspirations sont différents. On ne peut donc plus l'expliquer comme dans le modèle 1 par le chemin différent. Les tests prennent le même chemin malgré le nombre de solutions taboues différents ! En fait, il est clair que ces solutions taboues sont comptabilisées mais ne sont ni aspirées, ni choisies dans un des quelconques tests effectués comme meilleure solution d'un voisinage. Donc, celles-ci n'influencent en rien le chemin parcouru. D'autres tests contrediront très certainement ce phénomène qui est tout à fait dû au hasard.

Nous constatons aussi que le jeu des conditions taboues et de la fonction d'aspiration joue plus dans le deuxième type que dans le premier. Ceci est toujours dû à la sévérité plus grande du premier test par rapport au deuxième. Mais malgré tout, les résultats sont les mêmes. Ce qui ne veut pas dire que les conditions taboues ne servent à rien ! En effet, il ne faut pas oublier leur rôles dans l'élimination de bouclage dû au fait qu'on l'on peut choisir une solution moins bonne.

Nous pouvons encore faire une autre constatation du point de vue de l'influence de la longueur de la liste Tabou sur le temps d'exécution. Il apparaît que celui-ci n'est que doublé entre une liste de longueur 2 et 5000 alors que dans le modèle 1, il était multiplié par 10 au moins. Lorsque l'on génère notre voisinage dans notre modèle 2, il est beaucoup plus restreint à cause dans la contrainte qui vise à garder la longueur du tronçon fixe. Même si au début nous avons les nuances pour lesquelles le stock devient négatif au cours de la production, qui viennent s'ajouter aux nuances à essayer, celles-ci n'interviennent que tout au plus pendant 1/3 du temps d'exécution. Nous pouvons donc dire que le voisinage généré est beaucoup plus petit. Donc, il y a moins de test pour voir si une solution est taboue, et, donc, l'influence de la longueur de la liste Tabou sur le temps d'exécution est beaucoup plus faible.

Il nous reste à parler de la fonction d'aspiration. D'après les résultats, celle-ci ne joue pas du tout. Vu que celle qui a été implémentée est assez simple (aspiration aisée) et qu'elle ne joue pas, nous ne voyons pas l'intérêt de la modifier pour tester son influence.

I. Passage en revue des éléments du Tabou appliqués au modèle 2.

Suite au développement de notre algorithme pour ce deuxième modèle, des éléments ont été modifiés pour mieux l'adapter à ce problème. C'est pourquoi nous allons décrire briévement les éléments principaux du Tabou.

a. Les solutions.

S = ensemble des solutions au sens du modèle 1.

S_2 = ensemble des solutions au sens du modèle 2.

= (s , stock , pénalité)

où :

1) $s \in S$

2) "stock" est le stock final après la production de la coulée continue décrite dans "s" et la production au train.

3) "pénalité" est le nombre de fois que le stock d'une nuance devient négatif.

b. Les règles de modifications des solutions ou mouvements.

Nous exprimons cela suivant le voisinage généré qui est le suivant :

$N(s) = \{ s' \mid \exists k \in [1,n-2]$ tel que :

$\forall i \neq k, k+1 : (c_i,N_i) = (c'_i,N'_i)$ (1)

$N'_k , N'_{k+1} \in$ Nuances_en_face (2)

$c'_k \in [borne\_inf_{N'_k}, borne\_sup_{N'_k}]$ (3)

$c'_{k+1} \in [borne\_inf_{N'_{k+1}}, borne\_sup_{N'_{k+1}}]$ (4)

$c'_k + c'_{k+1} = c_k + c_{k+1}$ (5)

La condition (5) exprime le fait que la longueur du tronçon (en terme de nombre de coulées) doit rester constante au cours d'une modification.

c. La fonction d'évaluation.

L'expression de la fonction d'évaluation est la suivante :

fonction objective = enfournement à chaud - #pénalité . valeur_pénalité

où le nombre "valeur_pénalité" a été fixé à la valeur 500

d. Les conditions taboues.

Les conditions taboues ont évoluées. En effet, nous n'avons plus que des tronçons de deux séquences de coulées. Donc, un mouvement est tabou si :

$\exists$ t,t' $\in$ T où T est la liste Tabou tels que :

1) t = (i,c,N) et t' = (i',c',N')

2) i = k et i' = k+1

3) nuance_k (s') = N et nuance_{k+1} (s') = N'

4) nbre_coulées_k (s') = c et $\text{nbre\_coulées}_{k+1}$ (s') = c'

où s' est la solution sur laquelle on fait la modification.

e. La fonction d'aspiration.

Il n'y pas non pas de changement dans la définition de la fonction d'aspiration. Elle est donc un seuil que la fonction objective prise en la solution taboue doit franchir pour lever le statut tabou.

f. Nombre de détériorations successives avant de s'arrêter.

Nous avons gardé la valeur 200 pendant notre développement. Une valeur plus grande nous donnerait peut-être de meilleures solutions. Il pourrait être envisagé de prendre une valeur plus grande pour une implémentation réelle où il y aurait la possibilité d'arrêter l'algorithme à tout moment.

J. Ingrédients du modèle 2.

a. Construction d'une solution initiale.

Nous sommes partis dans nos deux modèles de solutions initiales quelconques, choisies aléatoirement mais en respectant les contraintes des bornes min-max des nombres de coulées. Ce type de solution initiale n'avait guère d'importance dans notre modèle 1 où l'algorithme partait d'un enfournement à chaud faible et en construisait un meilleur.

Dans notre second modèle (modèle avec stock), la solution initiale a plus d'importance car non seulement elle nous donne un enfournement à chaud faible mais en plus le nombre de pénuries est très important. Le fait que ce nombre soit grand, influence aussi la rapidité de l'algorithme car une itération prend plus de temps (le voisinage est plus grand).

Il faudrait alors peut-être construire une meilleure solution initiale. De cette façon, l'algorithme partirait d'une meilleure solution pour construire une solution acceptable de manière beaucoup plus rapide. Néanmoins, il y a un risque de construire une solution qui ne puisse plus être améliorée ou très peu. En effet, nous avons vu que résoudre rapidement (avec une grande pénalité) les pénuries, peut nous amener à de moins bons résultats. Le deuxième ingrédient pourrait peut-être améliorer cela.

b. Faire plusieurs essais .

Nous avons déjà mis ce principe dans les ingrédients du modèle 1 mais il nous semble qu'ici, il serait très intéressant de faire plusieurs essais si on utilise l'ingrédient qui vise à essayer plusieurs solutions initiales. En effet, on supprimerait le risque que nous avons mentionné ci-dessus.

Ces essais pourraient se faire en changeant le générateur de nombres aléatoires et en changeant la solution initiale. Il nous faudrait alors plusieurs moyens de créer une solution initiale avec un nombre de pénuries pas trop élevé.

c. Améliorer la fonction d'aspiration.

Nous pouvons peut-être améliorer la fonction d'aspiration en revoyant sa définition. La première définition que nous donnons est la suivante : on lève le statut tabou d'une solution lorsque la fonction d'évaluation prise en cette solution donne un résultat "une fois et un petit quelque chose " meilleur que le résultat de la meilleure solution actuelle. On pourrait exprimer cela par : $A^* = f(s^*).(1+n)$ où A^* est le niveau d'aspiration.

Une deuxième définition double cette fonction. On aurait donc deux niveaux d'aspiration. Le premier est A* = f(s*) comme nous l'avons employé dans nos modèles. Le second niveau est qu'on lève le statut tabou d'une solution si elle diminue le nombre de pénuries.

d. Favoriser les endroits où les pénuries peuvent être résolues.

Nous savons que les endroits où les pénuries peuvent être résolues sont dans le début de la coulée continue. En effet, le fait de couler une nuance au début peut éviter d'avoir des pénuries de celle-ci à la fin.

Il nous faudrait alors redéfinir la fonction qui définit l'emplacement du tronçon en une fonction qui favorise ces endroits. Une façon simple de créer cette fonction est la suivante : on fait générer plusieurs emplacements, par exemple 10, et l'on choisit le plus petit d'entre eux.

K. Conclusions sur le Tabou après le modèle 2.

a. Résultats proprement dits de l'algorithme.

Nous avons de quoi être satisfait de notre algorithme. En effet, la dernière version de notre algorithme nous trouve à chaque fois une solution acceptable et mieux encore, dans les 7 premiers tests, la solution est celle obtenue avec un stock infini. Les temps d'exécution sont satisfaisants bien que, en général, supérieurs aux temps des tests respectifs du modèle 1. Cela est tout à fait normal car des éléments en plus, tel que le calcul du stock, y sont ajoutés. En général, des temps de 3 à 4 minutes suffisent pour exécuter complètement l'algorithme.

La méthode du Tabou, comme certaines autres méthodes d'optimisation, a le défaut d'être fonction de la structure des solutions que l'on définit. Mais une fois celle-ci trouvée, quelques retouches et un peu d'expérimentation sont nécessaires pour arriver à de bons résultats. Néanmoins, ceci se réalise très simplement.

b. Les conditions taboues.

Il faudrait tirer les conclusions sur deux lignes. D'une part, nous avons le système des conditions taboues à juger et d'autre part, le système que l'on a implémenté.

Nous avons pu voir que les conditions taboues et la longueur de sa liste n'avaient que peu d'influence sur le résultat. Cela est peut-être dû au fait qu'elles ne jouent pas beaucoup dans ce problème. Mais, le peu qu'elles jouent est peut-être essentiel !

Nous avons modifié les conditions taboues en les rendant un peu moins sévère et nous avons pu voir qu'il n'y avait aucune différence aussi bien du point de vue résultat que du point de vue chemin parcouru (nombre d'itérations).

c. La fonction d'aspiration.

Dans notre modèle 1, la fonction d'aspiration jouait très peu. Maintenant, elle ne joue plus du tout mais c'est peut-être encore dû à la particularité de notre problème !

CHAPITRE IV : CONCLUSIONS SUR LA METHODE DU TABOU

La méthode du Tabou contient un certain nombre d'éléments principaux tels les modifications, la fonction d'évaluation,etc. Nous les avons décrits au début et à la fin du développement de nos deux modèles car ces derniers pouvaient entrainer leurs évolutions en vue de mieux adapter la méthode du Tabou à notre problème. Le développement nous a appris un certain nombre de choses à leur sujet; c'est ce que nous allons expliquer dans ce chapitre.

La structure de données de solutions et des modifications.

Dans les sections consacrées au développement de nos modèles, vous avez pu constater que ce sont sur les modifications ou mouvements que nous avons travaillé le plus. En effet, lors de l'étude des résultats correspondants au développement, nous avons pu voir que la structure de ces modifications influençaient fortement le comportement de notre algorithme, en tous ces aspects (résultats, temps d'exécution,...). Ces modifications sont donc très influentes et nous avons dû faire une étude très sérieuse pour que notre algorithme soit aussi performant.

Les modifications, de par leur définition, détermine le voisinage généré. Nous avons vu que la qualité de notre algorithme était déterminé par la taille et la nature du voisinage dans un sens donné par la structure des modifications. En effet, deux versions avec un grand voisinage peuvent donner des résultats très différents (voir modèle 1 version 1 et 7).

Nous avons donc vu que la méthode de génération du voisinage que nous avons implémentée, avait variée de nombreuses fois au cours de notre développement. Ces changements se font assez "facilement". En effet, la structure de notre algorithme "réunit" tout ce qui concerne les modifications dans une seule procédure (procédure de génération du voisinage) qui est donc la seule partie à modifier lors d'un changement de structure des modifications.

La fonction d'évaluation.

Nous avons vu que la majorité du temps est consacrée au calcul de la fonction objective. On se doit donc de l'optimiser au maximum pour éviter un gaspillage de temps. C'est pourquoi nous avons rassemblé le calcul de l'enfournement à chaud et le calcul du stock. On a pu aussi constater le lien étroit qui existait entre modification et temps de calcul de la fonction objective. Lorsque nous gardions notre longueur de tronçon (en nombre de coulées) fixe, seul le calcul de l'enfournement à chaud sur le tronçon devait être fait. Il en résultait une diminution considérable du temps d'exécution. Malheureusement, notre problème

faisait intervenir le stock qui nous a contraint à faire ce calcul sur toute la coulée continue.

Nous avons pu observer que la forme de la courbe de la fonction objective avait l'allure souhaitée c'est-à-dire que durant les toutes premières itérations, elle augmentait très fort et ensuite, elle contine à croitre plus faiblement. Cette forme de courbe "idéale" n'a été obtenue qu'avec les toutes dernières versions (modèle 1). Il apparaît donc que la forme de la courbe est une fonction de la structure des modifications.

Les conditions taboues.

Cet élément de la méthode du Tabou sert à éviter les minimums locaux. Quel que soit le modèle, nous avons pu voir que les conditions taboues ne "jouent" pas beaucoup. A cette situation, nous avons plusieurs manières de réagir. Quant à nous, nous pensons que le voisinage généré est suffisamment grand que pour éviter assez souvent ces minimums locaux. Et de ce fait, le piège ne se présente pas trop souvent.

Nous avons pu constater que la longueur de la liste n'influence pas le résultat. Comme les conditions ne jouent pas beaucoup, il est normal qu'elles n'influencent pas non plus le résultat. Quelle longueur de liste doit-on mettre ? Une trop grande liste a tendance à élever le temps d'exécution total. On aurait donc tendance à mettre une petite longueur de liste. Mais il ne faut pas oublier que ce peu d'influence est peut-être dû à la particularité de notre problème. Il ne peut donc y avoir de règle au sujet de la longueur de la liste. Dans la littérature, la taille de la liste varie en fonction du problème : 1, 2, 7 ou plus, ou encore une taille variable suivant certaines conditions.

La fonction d'aspiration.

Cet élément fait partie de la méthode du Tabou pour essayer de reprendre des solutions écartées par des conditions taboues illogiques. Comme les conditions taboues ne jouent pratiquement pas et que la fonction entre en jeu lorsqu'une solution est taboue, la fonction d'aspiration ne lève pratiquement jamais la caractère tabou d'une solution. De ce fait, nous ne saurions pas tirer de conclusion sur celle-ci.

Les conditions d'arrêt.

Nous avons utilisé le critère d'arrêt spécifié dans la méthode, à savoir : on arrête les itérations lorsqu'il n'y a pas d'amélioration constatée pendant "kmax" itérations. Ce paramètre évolue avec la structure des modifications. Ainsi, lorsque

nous avions pris une longueur de tronçon d'une séquence, l'algorithme se stabilisait très rapidemment et nous pouvions choisir une valeur assez faible pour ce paramètre.

Au cours de notre développement, nous avons dû constamment le réadapter. Pour fixer ce paramètre, nous avons d'abord choisi une valeur assez grande puis nous l'avons diminué de telle manière à ce qu'il n'y ait pas de changement de résultats.

Conclusion.

Si l'on se réfère aux résultats obtenus, nous pouvons alors dire que la méthode du Tabou est une bonne méthode d'optimisation. Elle donne de bons résultats en des temps tout à fait acceptable. Pour être sûr de cela, il nous faudrait de plus amples essais avec d'autres problèmes. Pour cela, nous pouvons nous référer à la littérature : aucuns des problèmes abordés ne semblent prendre en défauts la méthode.

Nous pensons que la réussite de cet algorithme est principalement dû à la méthode de parcours de l'ensemble des solutions : on prend la meilleure solution d'un voisinage qui ne comprend pas la solution avec laquelle on a généré ce voisinage. Ce principe dans lequel on accepte de détruire un résultat pour parcourir l'ensemble des solutions, ne se retrouve que dans peu de méthodes d'optimisation.

Si nous nous basons sur la mise en oeuvre de la méthode, nous pensons qu'avec un peu d'habitude et un peu d'expérimentation, on peut arriver très vite à de très bons résultats.

CHAPITRE V : LE MODELE 3

A. Enoncé input - output du problème du modèle 3.

Le modèle 3 ne faisant pas partie du cahier des charges, nous allons simplement tracer les grandes lignes de réalisation de celui-ci. L'énoncé du problème du modèle 3 est le suivant : étant donné des besoins en acier pour le train, trouver d'une part, l'ordre et les séquences de coulées de la coulée continue, et d'autre part, un ordonnancement des montages de profils et des tonnages au sein de ces derniers, tout en veillant à ce qu'il ne survienne aucune pénurie.

Dans le modèle 2, le train était une donnée du problème. Maintenant dans le modèle 3, c'est une donnée et un résultat du problème. En effet, en entrée, on donne un ensemble de montages de profils au sein desquels se trouve une série de tonnages à produire. Et en sortie, on doit fournir d'une part l'ordonnancement des montages de profils et d'autre part, l'ordonnancement des tonnages au sein de chaque montage de profils. Nous avons reçu des renseignements sur ces deux ordonnancements.

Pour l'ordonnancement des montages de profils, il existe des critères qui sont utilisés par la gestion commerciale de ProfilArbed. Ces critères sont très complexes car ils dépendent principalement des clients concernés par ces montages. C'est pourquoi il n'a jamais été envisagé d'automatiser cet ordonnancement.

Un couple (D,N) étant défini comme l'ensemble des postes de commande ayant même dérivé D et même nuance N, il existe 3 critères d'ordonnancement des tonnages au sein des montages de profils. Ceux-ci s'affrontent et leur importance est jugée par les planificateurs au vu du contenu des montages. Ces critères sont les suivants :

Critère 1 :

Rassembler les nuances (pour faciliter la tâche de la coulée continue et de l'aciérie).

Critère 2 :

Ordonnancer les dérivés pour "lisser" les productivités au train. Une heuristique serait d'alterner les dérivés à forte (300 tonnes par heure, par exemple) et à basse (100 tonnes par heure) productivité. De cette manière, on pourrait augmenter l'enfournement à chaud. En effet, lors du laminage à 300 tonnes par heure, il faut un déstockage constant élevé. Tandis que lors de laminage à 100 tonnes par heure, aucun déstockage n'est nécessaire.

Critère 3 :

Ordonnancer pour favoriser l'expédition directe c'est-à-dire sans passage par les parcs. Pour cela, il faut charger les poutrelles produites aussi tôt qu'elles sont laminées et donc, à aucun moment n'avoir plus de 6 destinations entamées (= pour lesquelles on a commencé à couper des poutrelles) puisqu'à la sortie, on dispose de 6 zones de tri avec emplacement de chargement d'un wagon.

Remarque : critère 4

Un dernier critère serait d'ordonnancer les couples (D,N) au lieu d'ordonnancer les dérivés D puis les nuances N (ou l'inverse). Ainsi, on se permet de ne rassembler ni les dérivés, ni les nuances N, au profit d'un savant mélange des deux.

Nous allons maintenant décrire notre troisième modèle sous ses input et ses outputs.

Input :

Le stock initial.

Une suite de nuances avec pour chacune d'elles, un stock limité, une borne inférieure et une borne supérieure (ces deux bornes sont les limites d'utilisation du répartiteur pour cette nuance d'acier), c'est-à-dire :

$$(\text{Nuance}_i , \text{Stock\_initial\_nuance}_i , \text{Borne\_inf}_i , \text{Borne\_sup}_i)_{1 \le i \le \text{Nbre\_nuance}}$$

où :

1) Nbre_nuance est le nombre de nuances différentes que la coulée continue peut fournir.

2) $\text{Stock\_initial\_nuance}_i < +\infty$, $\forall\, i : 1 \le i \le \text{Nbre\_nuance}$.

Le train.

Un ensemble de montages pour une semaine avec heure de début de laminage, nuance, tonnage et productivité , c'est-à-dire :

$$\text{Train} = \{ (\text{Tonnage}_i , \text{Productivité}_i , \text{Nuance}_i , \text{Heure de début}_i) \}_{1 \le i \le lg\_tr}$$

où "lg_tr" est le nombre total de tonnages à produire, chaque tonnage correspondant à une nuance et une productivité pour ce tonnage.

Ouput :

Le train.

Une séquence de montages pour une semaine avec nuance, tonnage et productivité, c'est-à-dire :

$$\text{Train} = \{ (\text{Tonnage}_i , \text{Productivité}_i , \text{Nuance}_i , \text{Heure de début}_i) \}_{1 \le i \le lg\_tr}$$

où "lg_tr" est le nombre total de tonnages à produire, chaque tonnage correspondant à une nuance et une productivité pour ce tonnage.

La coulée continue.

Une séquence de coulées à la coulée continue avec nuance et nombre de coulées, c'est-à-dire :

$$\text{C.C.} = (\text{Nombre\_coulées}_k , \text{Nuance}_k)\ 1 \le k \le Lg\_cc$$

où :

1) Lg_cc est la longueur de la liste des séquences de coulées.

2) $\text{Nombre\_coulées}_k \in [\ \text{Borne\_inf}_{\text{nuance}_k} , \text{Borne\_sup}_{\text{nuance}_k}\ ]$

Le stock.

Un stock pour chacune des nuances (stock obtenu à partir de "stock_initial_nuance" en fin de production du train et de la coulée continue) c'est-à-dire :

$$\text{Stock} = (\text{Stock\_final\_nuance}_k)\ 1 \leq k \leq \text{Nbre\_nuance}$$

où $0 \leq \text{stock\_final\_nuance}_k < +\infty$, $\forall\ k : 1 \leq k \leq \text{Nbre\_nuance}$

B. Passage en revue des éléments du Tabou appliqué au modèle 3.

Nous allons décrire les éléments principaux du Tabou. Il est clair que ces propositions pour les éléments principaux ont été considérés sur base de l'expérience acquise lors de la réalisation des deux premiers modèles. Bien entendu, un développement sera nécessaire.

a. Les solutions.

L'ensemble des solutions du modèle 3 n'est naturellement plus le même que celui du modèle 2. En effet, le train fait partie maintenant des résultats de notre algorithme.

S_2 = ensemble des solutions au sens du modèle 2.

S_3 = ensemble des solutions au sens du modèle 3.

= (s , tr)

où :

1) $s \in S_2$

2) "tr" est un ordonnancement du train.

b. Les règles de modification des solutions ou mouvements.

Nous allons faire nos modifications de la même manière que dans les autres modèles : nous choisissons au hasard un tronçon de deux séquences de coulées dans la coulée continue. Sur ce tronçon, nous allons faire toutes les modifications possibles, à savoir, les modifications sur la coulée continue et les modifications sur le train. Comme pour le modèle 2, nous adopterons la contrainte qui garde constante la somme des nombres de coulées des deux séquences de coulées. De cette manière, nous pouvons trouver un stock sans pénurie (voir modèle 2) et nous limitons le nombre de solutions voisines dues aux modifications de la coulée continue. Car il ne faut pas oublier que les modifications dues au train vont augmenter la taille de notre voisinage. Nous espérons toutefois que cette taille ne sera pas trop grande pour ne pas passer trop de temps sur une itération.

Le voisinage généré est le suivant :

$N(s) = \{ s' \mid \exists k \in [1,n-2]$ tel que :

$\forall i \neq k, k+1 : (c_i,N_i) = (c'_i,N'_i)$ (1)

$N'_k , N'_{k+1} \in$ Nuances_en_face (2)

$c'_k \in [\text{Borne\_inf}_{N'_k} , \text{Borne\_sup}_{N'_k}]$ (3)

$c'_{k+1} \in [\text{Borne\_inf}_{N'_{k+1}} , \text{Borne\_sup}_{N'_{k+1}}]$ (4)

$c'_k + c'_{k+1} = c_k + c_{k+1}$ (5)

({tonnages en face})' = Permutation ({tonnages en face}) (6) }

"{tonnages en face}" sont tous les tonnages à produire qui sont en face du tronçon considéré (voir Chapitre II § I). La condition (6) exprime le fait que tous les tonnages en face se retrouvent toujours en face suite à une modification mais qu'ils peuvent être permutés.

Si notre algorithme devait prendre trop de temps, nous pourrions diminuer celui-ci en réduisant la taille du voisinage généré en transformant la permutation des tonnages en face du tronçon, en deux permutations sur les tonnages en face des séquences de coulées respectivement. Par exemple, si nous avons 10 tonnages à produire en face de notre tronçon, nous aurons 10! permutations possibles. Tandis que, par exemple, 5! . 5! permutation (ou 6! . 4!) mais de toute façon < 10!, si nous decomposons la permutation en deux permutations sur les tonnages en face.

c. La fonction d'évaluation.

Cette fonction sera la même que celle du modèle 2 à laquelle nous ajouterons une pénalité fonction du critère d'ordonnancement que nous choisissons. L'expression de la fonction objective est la suivante :

fonction objective = enf. à chaud - #pénalité . pénalité1 - x . pénalité2

où :

1) "pénalité1" est la pénalité pour une pénurie (voir modèle 2).

2) "pénalité2" est la pénalité pour x où x est :

- pour le critère 1 : le nombre de changements de nuances.

- pour le critère 2 : le nombre de fois que deux tonnages à productivité faible (ou forte) se suivent.

Il est évident que ce nombre x ne peut pas être nul dans un cas comme dans l'autre sauf éventuellement pour quelques exceptions très rares. Pour le critère 1, plus ce nombre est petit et plus les nuances d'un même type sont groupées. Pour le critère 2, plus ce nombre est petit, plus on a un train où une tonnage à productivité faible suit un tonnage à productivité forte. La valeur de la pénalité "pénalité2" ne peut se fixer que par expérimentation comme nous l'avons fait pour la valeur de pénalité "pénalité1" au cours du développement de notre modèle 2.

d. Les conditions taboues.

Nous pensons qu'il faut reprendre les mêmes conditions taboues sans en ajouter ou modifier quoi que ce soit (par exemple une condition en rapport avec le train). En effet, nous avons deux bonnes raisons à cela. La première est qu'une condition taboue sur les tonnages devrait retenir l'ordre des tonnages qui se trouvent en face : cela peut prendre énormément de place en mémoire. La deuxième est que, comme on peut modifier les séquences de coulées de la coulée continue de une ou de deux séquences en plus ou en moins (suivant respect de la contrainte), cela provoque un décalage de la coulée continue par rapport au train : la condition taboue peut ne plus avoir aucune incidence puisqu'elle ne pourrait plus "jouer" que si c'est le même tronçon (même heure de début et de fin). Il est évident qu'il existe d'autres conditions taboues très sophistiquées mais elles pourraient entrainer un temps important à la mise à jour de la liste Tabou et au test d'existence d'une condition dans cette liste.

e. La fonction d'aspiration.

Nous reprendrons la même définition que dans le modèle 2, à savoir qu'on peut lever le statut tabou à une solution taboue si elle donne un meilleur résultat que l'optimum intermédiaire.

Bibliographie

[1] " Sidérurgie : produits longs; les investissements se concrétisent" , Périodique d'entreprise du groupe ARBED, septembre 1993.

[2] Peat MARWICK , "Rapport sur l'exercice 1992 du groupe ARBED", 1992.

[3] D. de WERRA et A. HERTZ , " Tabu Search Techniques : A tutorial and an application to neutral networks" , Springer Verlag , vol.11, pp.131-141, 1989.

[4] D. de MUYNCK, " Méthode 'Tabu search' : Technique et applications" , Systèmes et modèles Sidmar N.V., avril 1992.

[5] A. HERTZ , " Tabu search for large scale timetabling problems " , European Journal of Operational Research , vol.54, pp.39-47, 1991.

[6] C. FRIDEN, A. HERTZ et D. de WERRA , " Tabaris : an exact algorithm based on tabu search for finding a maximum independent set in a graph " , Pergamon Press , vol.17, pp. 437-445,1990.

[7] C. FRIDEN, A. HERTZ et D. de WERRA, " STABULUS : A technique for Finding Stable Sets in Large Graphs with Tabu Search", Springer Verlag, vol.42, pp.35-44, 1989.

[8] F. GLOVER, " Tabu Search- Part I ", ORSA Journal on Computing , vol.1, pp.4-32, 1989.

[9] F. GLOVER, " Tabu search : a tutorial", Interfaces, vol.20, pp. 74-94, 1990.

[10] L. MERESSE et M. LESCRENIER, " Optimisation de la production du train de Grey de Differdange pour les installations futures liées à la coulée continue ", Service de l'informatique Unité Modélisations et Optimisations, Réf. : EX.TG.ML.ENON.02, 1993.

[11] M. LESCRENIER, " Optimisation de la production du train de Grey de Differdange pour les installations futures liées à la coulée continue ", Service de l'informatique Unité Modélisations et Optimisations, Réf. : EX.PR.LM.1406.01, 1993.

[12] M. LESCRENIER, " Optimisation de la production du train Grey de Differdange pour les installations futures liées à la coulée continue " , Service de l'informatique Unité Modélisations et Optimisations, Réf. : EX.CR.LM.1406.01, 1993.

Table des matières.

CHAPITRE I : INTRODUCTION.

CHAPITRE II : Le modèle 1

CHAPITRE III : Le modèle 2

CHAPITRE IV : Conclusions générales sur le Tabou

CHAPITRE V : Le modèle 3.